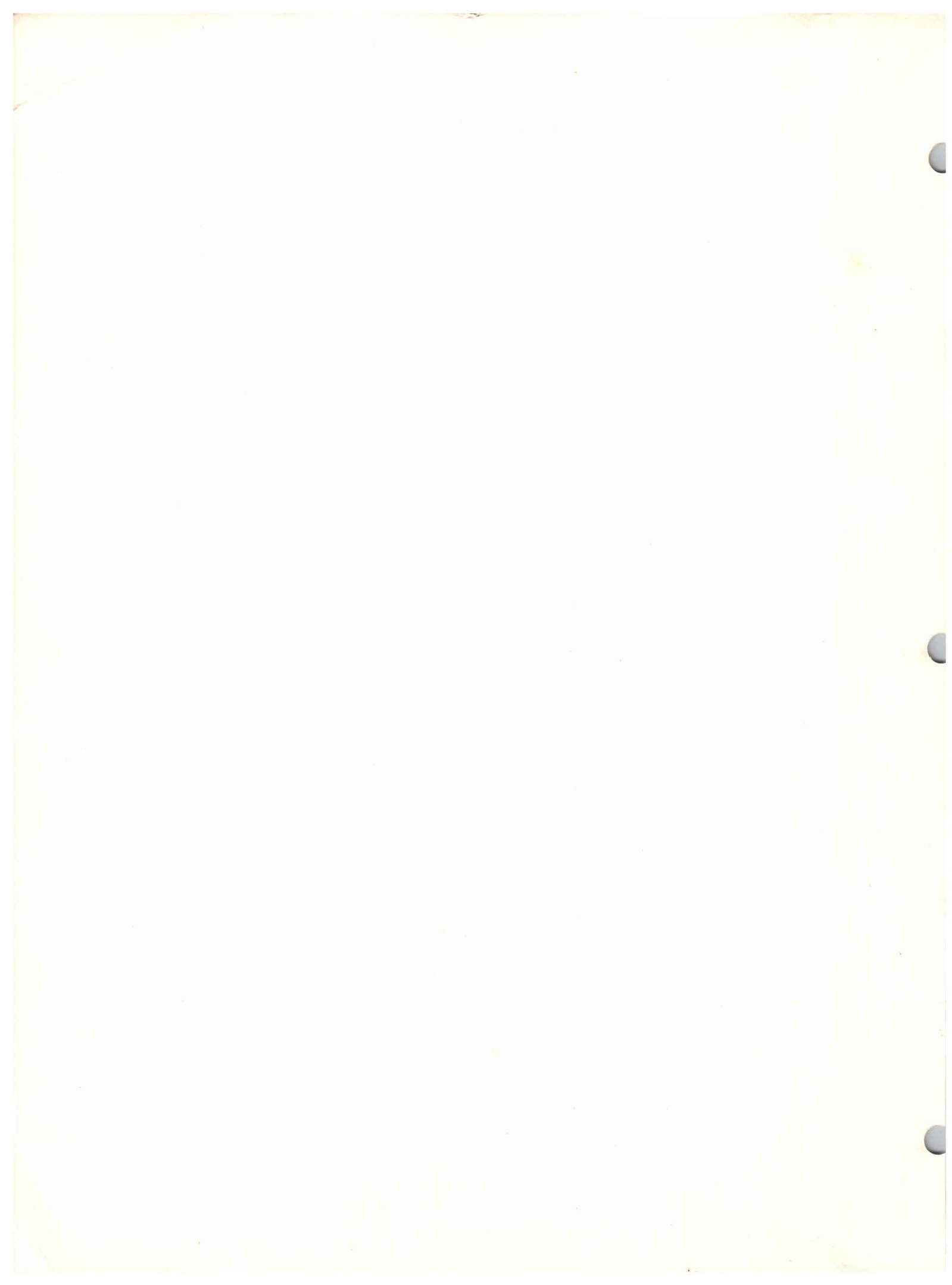


# HUG SOFTWARE

## VOLUME







## HUG SOFTWARE VOLUME II

NOTE: All BASIC programs in Volume II were prepared with Extended Benton Harbor BASIC 10.02 (Heath model #H8-13). Unless otherwise noted, all programs can be loaded by the command, LOAD "Program Name" or LOAD " ".

Some programs may convert to disk; however, it is the users' responsibility to make whatever modifications necessary.

All questions regarding an individual program should be directed to the author, whose name and address can be found in the Software Catalog and, in some cases, within the program itself.

Additional documentation on some of the programs may be found at the end of the listing.







# I N D E X

## Volume II

BASIC Programs 885-1012

<u>Tape Index Counter</u> <u>Side 1</u>	<u>Program Name</u>	<u>Page Number</u>
25	Data Base-Wielde (use 'FLOAD')	B-1
45	Create-A-Program-Dennison	B-8
85	I/O Routines-Masuda	B-21
100	File Maintenance-Roseman	B-26
115	Multisort-Roseman	B-32
125	4 Mini-Bug-Horne	B-38
145	Chi-Square-Forsythe	B-45
185	5 Expansion Chamber-Fuller	B-46
150	Multiple File Management-Hicks	B-54
205	Bolt Hole Locator	B-67
210	Calendar-Wilkinson	B-68
255	Resonant Freq/Calc-Borden	B-75
260	Pad Calculator-Borden	B-78
265	Index Leader-Craig	B-80
Side 2		
015	Weather Records-Myrup	B-81
035	'Get File for Weather'	
050	Low Pass Design-Dacey	B-87
065	Band Pass Design-Dacey	B-92
085	Amplifier Design-Dacey	B-98



# I N D E X

## Volume II

BASIC Programs 885-1012 Cont'd.

<u>Tape Index Counter</u> <u>Side 2</u>	<u>Program Name</u>	<u>Page Number</u>
100	RC OSC Design-Dacey	B-103
115	DECOCT-Kern	B-107
130	MORSCII-Kern (FLOAD)	B-113
145	BASICLOCK-Kern	B-126
155	Text Handler-Greenhalgh	B-128
170	Intelligent Disassembler-Warner	B-131
Side 1 225	Utility Graph-Fuller	B-137
Side 1 235	Electronic Formulas-Crabtree	B-141
Side 2 190	Metric/US Conversion-Stillman	B-149
202	BEEP TED8 Source Code	B-151
210	TIME TED8 Source Code	B-153

TED8 Source Code 885-1014

<u>Program Name</u>	<u>Page Number</u>
Renumber-Ignatius	1
Renumber/Merge-Moss	22
Utility Routines-Price	43
TAPE Management-Turner	70
8080 Disassembler-Shiffrin	108
CCS'SUPPRESS-Morgan	121
Mail Label-Farmer	130



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DATA BASE - WIEDLE

```
00001 REM    INTERACTIVE DATA BASE  (9/1/78)
00002 REM    RUSSELL WIEDLE AUTHOR
00003 REM    4717 WEST 30TH
00004 REM    TOPEKA,KS 66614
00005 REM    (913) 272-8351
00050 GOSUB 1500
00075 B$=CHR$(10)+CHR$(10)+CHR$(10)+CHR$(10)
00076 B$=B$+B$+B$
00095 IF M<>0 GOTO 320
00105 L9=10
00110 DIM L(L9), L5(L9)
00120 A9=55
00130 DIM A$(A9), A(A9,3)
00180 G$="=====
00200 M1=0
00210 D1=1
00220 S1=2
00230 M=1
00300 N=1
00310 LINE INPUT "SUBJECT ==> ";A$(1)
00315 L(0)=1
00320 GOSUB 8100
00380 GOTO 320
00400 REM ADD
00402 IF C9$="A" THEN P1=99
00403 IF C$="C" AND P1<1 THEN PRINT B$;"NEED LINE #":RETURN
00405 GOSUB 8000
00407 C2=0
00410 LINE INPUT "ENTER DATA ==> ";D$
00415 PRINT B$;D$;" **NEW DATA**"
00420 IF A(M,D1)=0 THEN GOTO 940
00430 F=1
00440 S=A(M,D1)
00450 C=S
00460 IF P1>=1 THEN GOTO 600
00480 C=0
00500 A(M,D1)=N
00520 P=-1
00540 GOTO 700
00600 FOR I=1 TO INT(P1)
00620 C=S
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DATA BASE - WIELDE < CONT'D >

```
00640 P=I
00660 S=A(C,S1)
00670 IF S=0 THEN GOTO 700
00680 NEXT I
00700 REM ADD OR REPLACE?
00710 IF C#="C" AND P1=P GOTO 840
00740 A(N,M1)=M
00780 A(N,S1)=S
00800 A(C,S1)=N
00805 A(0,S1)=0
00810 A$(N)=D$
00820 GOTO 900
00840 A$(C)=D$
00900 REM
00920 GOTO 1000
00940 A(N,M1)=M
00950 A(N,S1)=0
00980 A(M,D1)=N
00990 A$(N)=D$
00995 A(0,S1)=0
01000 RETURN
01050 REM SELECT
01060 IF P1<>INT(P1) THEN GOTO 1190
01065 T2=P1
01100 S=A(M,D1)
01110 S2=S
01120 FOR I=1 TO P1
01125 S2=C
01130 C=S
01140 S=A(C,S1)
01150 NEXT I
01155 IF C=0 THEN GOTO 1190
01160 C2=C
01170 PRINT B$
01180 GOTO 1199
01190 PRINT B$;"ENTER EXISTING #"
01195 C2=0
01199 RETURN
01200 REM INSERT-SELECTED
01203 GOSUB 1700
01205 IF C2=0 THEN GOTO 1299
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DATA BASE - WIELDE < CONT'D >

```
01210 N=C2
01220 P1=P1+.01
01230 IF A(A(N,M1),D1)=N THEN A(A(N,M1),D1)=A(N,S1)
01235 IF A(A(N,M1),D1)<>N THEN A(S2,S1)=A(N,S1)
01240 D#=A$(N)
01250 GOSUB 420
01260 C2=0
01270 PRINT B$;D$;" **INSERTED**"
01299 RETURN
01300 REM DELETE
01305 T1=P1
01306 P1=T2
01310 IF C2=0 THEN 1499
01330 N=C2
01340 M2=A(N,M1)
01345 C3=A(N,D1)
01350 IF A(N,D1)=0 THEN GOTO 1390
01355 A(C3,M1)=M2
01358 C4=C3
01360 C3=A(C3,S1)
01365 IF C3<>0 THEN GOTO 1355
01370 A(C4,S1)=A(N,S1)
01380 A(N,S1)=A(N,D1)
01390 IF A(A(N,M1),D1)<>N THEN A(S2,S1)=A(N,S1)
01400 IF A(A(N,M1),D1)=N THEN A(A(N,M1),D1)=A(N,S1)
01410 A(N,M1)=0
01420 A(N,D1)=0
01430 A(N,S1)=0
01440 PRINT B$;A$(N);" **DELETED**"
01445 C2=0
01450 A$(N)=" "
01460 A(0,S1)=0
01465 IF T1<2 GOTO 1499
01468 PAUSE 800
01470 T1=T1-1:GOSUB 1050
01475 GOTO 1310
01499 RETURN
01500 REM HELP
01510 PRINT "COMMANDS:"
01530 PRINT "(A..) ADD ITEM"
01550 PRINT "(C..) CHANGE"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DATA BASE - WIELDE < CONT'D >

```
01570 PRINT "(L..) GOTO LOWER LEVEL"
01590 PRINT "(H..) GOTO HIGHER LEVEL"
01610 PRINT "(T) GOTO TOP"
01630 PRINT "(S..) SELECT FOR MOVE/DELETE"
01650 PRINT "(I..) INSERT SELECTED ITEM"
01670 PRINT "(X..) DELETE SELECTED ITEM      (O)      DISPLAY OUTLINE
"
01680 PRINT "(R) RE-DISPLAY ITEMS          (F1)     FIND FIRST MATC
H"
01690 PRINT "(?) LIST COMMANDS            (F)      FIND NEXT MATC
H"
01699 RETURN
01700 REM VALIDATE HIER
01710 N=M
01720 FOR I=0 TO L
01730 IF A(N,M1)=C2 THEN GOTO 1770
01740 N=A(N,M1)
01750 NEXT I
01760 RETURN
01770 PRINT "INVALID HIERARCHY" : C2=0
01790 RETURN
02000 REM FIND/OUTLINE
02005 F5=0
02010 IF C$="O" THEN P1=1:S$=CHR$(0):PRINT B$;" ";A$(1):GOTO 2070
02020 LINE INPUT "DATA TO BE FOUND ==> ";S$
02040 IF LEN(S$)=0 THEN S$=S2$
02050 S2$=S$
02060 IF P1=0 THEN P1=2
02070 IF C$="F" THEN PRINT B$
02080 IF P1=1 GOTO 2160
02100 M5=M:L5=L:U5=M
02120 GOTO 2200
02160 M5=1:L5=0:U5=1
02200 FOR I=0 TO L9
02220 L5(I)=L(I)
02240 NEXT I
02250 CNTRL 4,0
02260 GOSUB 2400
02270 CNTRL 4,1
02271 IF C$="O" THEN PAUSE 2000:PRINT :PRINT :GOTO 2399
02272 PRINT "<<"S$;">> ";
```



<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: DATA BASE - WIELDE < CONT'D >

```
02280 IF F5=0 THEN PRINT "NOT FOUND":GOTO 2399
02300 PRINT "DATA FOUND"
02320 FOR I=0 TO L9
02340 L(I)=L5(I)
02350 NEXT I
02360 M=V5
02380 L=L5
02399 RETURN
02400 REM -HIERFIND-
02410 L5(L5)=M5
02420 V5=A(M5,D1)
02430 L5=L5+1
02440 L5(L5)=V5
02450 IF V5<>0 AND F5=0 GOTO 2520
02460 L5=L5-1
02470 IF L5=0 GOTO 2700
02480 V5=L5(L5)
02490 M5=A(V5,S1)
02500 V5=M5
02505 L5(L5)=V5
02510 GOTO 2700
02520 REM
02521 IF C#="0" THEN PRINT TAB(L5*2);A$(V5):GOTO 2560
02525 PRINT CHR$(7)
02528 FOR I=1 TO LEN(A$(V5))-LEN(S#)+1
02530 IF (MID$(A$(V5),I,1)=LEFT$(S#,1))=65535 THEN F5=(MID$(A$(V5),I,LEN(S#))=S#)
02540 IF F5<>0 THEN GOTO 2700
02550 NEXT I
02560 IF A(V5,D1)<>0 GOTO 2650
02570 IF A(V5,S1)<>0 GOTO 2610
02580 V5=0
02590 GOTO 2700
02610 V5=A(V5,S1)
02620 L5(L5)=V5
02630 GOTO 2700
02650 V5=A(V5,D1)
02660 L5=L5+1
02670 L5(L5)=V5
02700 REM
02705 IF L5=0 GOTO 2740
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DATA BASE - WIELDE < CONT'D >

```
02710 IF F5<>0 GOTO 2740
02720 GOTO 2450
02740 RETURN
07000 REM PRINT RTN
07010 GOSUB 8700
07015 X2=1
07020 Q$=""
07025 X=A(M,D1)
07026 IF A(X,D1)<>0 THEN Q$=""+"
07030 PRINT X2;CHR$(8);Q$;A$(X)
07035 IF (X2+L)>9 THEN PAUSE (X2+L)*75
07040 X2=X2+1
07045 Q$=""
07050 X=A(X,S1)
07060 IF X<>0 THEN GOTO 7026
07999 RETURN
08000 REM FIND SLOT
08010 FOR N2=MAX(2,N2) TO A9
08020 IF A(N2,M1)=0 GOTO 8050
08030 NEXT N2
08035 N2=2
08040 PRINT "FULL!!"
08045 GOTO 320
08050 N=N2
08055 IF N2<(A9-10) GOTO 8090
08060 PRINT A9-N2;"MORE ADDS POSSIBLE"
08070 N2=2
08090 RETURN
08100 REM INPUT PROCESSOR
08110 LINE INPUT "COMMAND ==> ";C$
08111 IF LEN(C$)>3 THEN 8114
08112 IF ASC(C$)>62 AND ASC(MID$(C$,2))<58 THEN 8120
08114 PRINT CHR$(7);CHR$(7);CHR$(7);CHR$(7);CHR$(7)
08116 GOTO 8110
08120 P1=VAL(RIGHT$(C$,LEN(C$)-1))
08125 C9=C$
08130 C$=LEFT$(C$,1)
08140 IF C$="L" THEN GOSUB 8500
08150 IF C$="H" THEN GOSUB 8600
08160 IF C$="A" OR C$="C" THEN GOSUB 400
08170 IF C$="T" THEN GOSUB 8800
```

<H><U><G>    <S><O><F><T><W><A><R><E>    <U><O><L>    II

```

08180 IF C$="S" THEN GOSUB 1050
08185 IF C$="F" OR C$="O" THEN GOSUB 2000
08190 IF C$="I" THEN GOSUB 1200
08195 IF C$="X" THEN GOSUB 1300
08200 IF C$="?" THEN GOSUB 1500
08220 IF C$="R" THEN PRINT B$
08280 IF C$<>"?" THEN GOSUB 7000
08290 RETURN
08500 REM LOWER
08505 PRINT B$
08510 S=A(M,D1)
08515 IF P1<>INT(P1) THEN GOTO 8560
08518 IF P1=1 THEN GOTO 8540
08520 FOR I=1 TO P1-1
08525 S=A(S,S1)
08535 NEXT I
08540 IF S=0 THEN GOTO 8560
08550 L=L+1
08552 L(L)=S
08555 M=S
08560 RETURN
08600 REM HIGHER
08605 PRINT B$
08610 IF L=0 THEN GOTO 8680
08620 L=MAX(0,(L-MAX(P1,1)))
08640 M=L(L)
08680 RETURN
08700 REM PRINT HIER
08705 IF C2<>0 THEN PRINT A$(C2); " **SELECTED**"
08710 PRINT G$
08720 FOR I=0 TO L
08730 PRINT TAB(I);A$(L(I))
08750 NEXT I
08760 PRINT G$
08799 RETURN
08800 REM TOP
08805 PRINT B$
08810 M=1
08820 L=0
08830 RETURN

```



## CONTENTS

OVERVIEW	.	.	.	.	.	.	1
COMMANDS	.	.	.	.	.	.	2
SAMPLE DATA BASE	.	.	.	.	.	.	4
SAMPLE PROGRAM	.	.	.	.	.	.	5
HELP	.	.	.	.	.	.	5
LOWER	.	.	.	.	.	.	5
CHANGE	.	.	.	.	.	.	6
ADD	.	.	.	.	.	.	7
HIGHER	.	.	.	.	.	.	7
FIND	.	.	.	.	.	.	8
TOP	.	.	.	.	.	.	8
REDISPLAY	.	.	.	.	.	.	9
SELECT	.	.	.	.	.	.	9
INSERT (MOVE)	.	.	.	.	.	.	9
DELETE	.	.	.	.	.	.	10
OUTLINE	.	.	.	.	.	.	11
STORAGE REQUIREMENTS	.	.	.	.	.	.	12
OTHER APPLICATIONS	.	.	.	.	.	.	13
SUBROUTINES	.	.	.	.	.	.	15
MODIFICATIONS	.	.	.	.	.	.	16
VARIABLES	.	.	.	.	.	.	17
INTERNAL STRUCTURE	.	.	.	.	.	.	18
CHANGING DATA BASE SIZE	.	.	.	.	.	.	19

## OVERVIEW

Interactive Data Base (IDB) is a data storage and retrieval system. The data base has a hierarchical organization which allows data to be added, changed, or retrieved with a minimum of effort. The hierarchy can be used in any way desired to store data. For many applications the highest level of the hierarchy will be used to identify the data base and the next level is used to break the data into several logical categories (usually 2-10) that will be beneficial in retrieving the data. The third level in this case could be the data or sub-categories. The data when entered can be single data item or a hierarchy of data items.

Data retrieval is accomplished by "navigating" up, down, and around the hierarchy with a set of commands that are provided with IDB. After each command is executed, data items at the current position are displayed along with all of the data items in the direct path from the top of the hierarchy down to the current position. A "Find" command can be used to locate a data item containing a matching character string. The "Find" command can be used to search data items hierarchically forward from the current position or from the beginning of the hierarchy. The data base may also be listed in an indented outline format if desired. If a hardcopy device is available, the outline format makes a good working document when the computer is not available.

The data base is maintained by commands that add data items, change data items, delete data items, and rearrange data items within the hierarchy.

## COMMANDS

This section describes the commands that are provided with IDB. The command syntax rules are:

- 1.) First character must be alphabetic or "?".
- 2.) Second and third character must be numeric if allowed (indicated by "..")
- 3.) Commands that require text type data will prompt for it (E.G. ADD, CHANGE, FIND)

### (A..) ADD ITEM

"ADD" will add a data item to the list currently displayed. If a number is entered with the add command the new item will be added after the item specified. To add a data item as the first in the list specify "A0". If no number is specified the item will be added to the end of the list.

### (C..) CHANGE ITEM

"CHANGE" will allow the text of the specified item to be replaced.

### (L..) GOTO LOWER LEVEL

"LOWER" will move the current position within the hierarchy one level lower on the specified path (data item #).

### (H..) GOTO HIGHER LEVEL

"HIGHER" will move the current position within the hierarchy up the path currently displayed by the number of levels specified. The default is one.

### (T) TOP

"TOP" will move the current position within the hierarchy to the highest level.

### (S..) SELECT FOR MOVE/DELETE

"SELECT" is used to identify a data item that is to be moved to another location within the hierarchy (See (I..) "INSERT") or to identify a data item to be deleted (See (X) "DELETE").

### (I..) INSERT SELECTED ITEM

"INSERT" is basically the same as "ADD" but the new data is the data item that was selected with "SELECT".



(X) DELETE SELECTED ITEM

"DELETE" will delete the data item that has been selected with the "SELECT" command. When the the data item is deleted any items at a lower level below the deleted item will move up one level. If a number is specified with the delete command (e.g. X12) that number of data items will be deleted from the hierarchy starting with the data item selected. The multiply delete feature may be disabled if so desired (see section on modifications).

(R) RE-DISPLAY ITEMS

"REDISPLAY" will present the last display of data items again. This command is used if number of data items is greater than screen size on a CRT terminal and a redisplay of the first items is desired.

(?) LIST COMMANDS

"?" is the help function. A list of the available commands is displayed for quick reference.

(O) DISPLAY OUTLINE

"OUTLINE" will display the entire data base in outline format with the data items for each level indented below the owner of those data items on the previous level.

(F1) FIND FIRST MATCH

"FIND FIRST" will prompt for a character string and then starting at the "TOP" searches the data base hierarchically stopping at the first match.

(F) FIND NEXT MATCH

"FIND NEXT" will prompt for a character string (or will use the last string entered if "return" is hit with no character string) and then starting with data item number one of the current display a hierarchical search will be performed stopping at the next match in the data base.

```

:PIGS
:SMITH  --:ANIMALS  --:SHEEP
:BOSSY
:MOLLY
:HECTOR
:HORSES
:SMITH  --:COWS  --:BOSSY
:MOLLY
:HECTOR
:HORSES
:SMITH  --:CORN  --:BOSSY
:BEANS
:SMITH  --:DUCKS  --:200 IN NORTH SHED
:50 IN SOUTH SHED
:CHICKENS
:GEESE
:SMITH  --:TRACTORS  --:JOHN DEERE
:FARMALL
:SMITH  --:EQUIPMENT  --:77 FORD
:76 DODGE
:SMITH  --:RABBITS
:CHICKENS
:SMITH  --:SHEEP
:SMITH  --:BARLEY  --:7/7/78 HAIL DAMAGE
:8/1/78 2 TONS TO COORS
:RYE
:SMITH  --:CROPS  --:JOB: SHEPHERD
:ADDRESS
:PHONE (913) 555-1212
:TOPEKA, KS.
:SMITH  --:LUBRICANTS  --:10W30 (MOBIL)
:WD40
:SMITH  --:R2D2  --:IN CASE OF ACCIDENT NOTIFY C3P0

```

# SAMPLE PROGRAM

```
*LOAD "IDB"  
*GET "FARM DATA BASE"  
*CONTINUE
```

```
COMMAND =====)  ?
```

## COMMANDS:

(A..)	ADD ITEM	
(C..)	CHANGE ITEM	
(L..)	GOTO LOWER LEVEL	
(H..)	GOTO HIGHER LEVEL	
(T)	GOTO TOP	
(S..)	SELECT FOR MOVE/DELETE	
(I..)	INSERT SELECTED ITEM	
(X)	DELETE SELECTED ITEM	(O) DISPLAY OUTLINE
(R)	RE-DISPLAY ITEMS	(F1) FIND FIRST MATCH
(?)	LIST COMMANDS	(F) FIND NEXT MATCH

```
COMMAND =====)  T
```



-----  
FARMS  
-----

1)+ SMITH  
2)+ JONES  
3)+ CLARK  
COMMAND =====) L1

-----  
FARMS  
SMITH  
-----

1)+ ANIMALS  
2)+ CROPS  
COMMAND =====) L2

-----  
FARMS  
SMITH  
CROPS  
-----

1) CORN  
2) BEANS  
COMMAND =====) C2  
ENTER DATA =====) SOYBEANS

SOYBEANS \*\*NEW DATA\*\*

-----  
FARMS

SMITH

CROPS  
-----

1) CORN

2) SOYBEANS

COMMAND =====) A1

ENTER DATA =====) WHEAT

WHEAT \*\*NEW DATA\*\*

-----  
FARMS

SMITH

CROPS  
-----

1) CORN

2) WHEAT

3) SOYBEANS

COMMAND =====) H1

-----  
FARMS  
SMITH  
-----

1)+ ANIMALS  
2)+ CROPS  
COMMAND =====) F1

DATA TO BE FOUND =====) BARLEY

((BARLEY)) DATA FOUND  
-----

FARMS  
CLARK  
CROPS  
BARLEY  
-----

1) 7/7/78 HAIL DAMAGE  
2) 8/1/78 2 TONS TO COORS  
COMMAND =====) F

DATA TO BE FOUND =====) BARLEY

((BARLEY)) DATA NOT FOUND  
-----

FARMS  
CLARK  
CROPS  
BARLEY  
-----

1) 7/7/78 HAIL DAMAGE  
2) 8/1/78 2 TONS TO COORS  
COMMAND =====) T

-----  
FARMS  
-----

1)+ SMITH  
2)+ JONES  
3)+ CLARK  
COMMAND =====) R

-----  
FARMS  
-----

1)+ SMITH  
2)+ JONES  
3)+ CLARK  
COMMAND =====) S3

CLARK \*\*SELECTED\*\*

-----  
FARMS  
-----

1)+ SMITH  
2)+ JONES  
3)+ CLARK  
COMMAND =====) IØ



CLARK \*\*INSERTED\*\*

-----  
FARMS  
-----

1)+ CLARK

2)+ SMITH

3)+ JONES

COMMAND =====) F1

DATA TO BE FOUND =====) TRU

((TRU)) DATA FOUND  
-----

FARMS

JONES

EQUIPMENT

TRUCKS  
-----

1) 77 FORD

2) 76 DODGE

COMMAND =====) S2

76 DODGE \*\*SELECTED\*\*  
-----

FARMS

JONES

EQUIPMENT

TRUCKS  
-----

1) 77 FORD

2) 76 DODGE

COMMAND =====) X

76 DODGE \*\*DELETED\*\*  
-----

FARMS

JONES

EQUIPMENT

TRUCKS  
-----

1) 77 FORD

COMMAND =====) O

FARMS

SMITH

ANIMALS

PIGS

SHEEP

COWS

BOSSY

MOLLY

HECTOR

HORSES

CROPS

CORN

WHEAT

SOYBEANS

JONES

ANIMALS

DUCKS

200 IN NORTH SHED

50 IN SOUTH SHED

CHICKENS

GEESE

EQUIPMENT

.  
.  
.

COMMAND =====)

## STORAGE REQUIREMENTS

IDB as distributed has the following requirements;

Program Text	3973*
Symbols	1356 (50 Data Items)
For Loops	24
Subroutine Calls	8
Strings	615 (Sample Data Base -- 49 Items)
	<u>5976</u>

\* Program text may be reduced by deleting the following statements;

<u>STATEMENT</u>	<u>BYTES</u>	<u>FUNCTION</u>
1-5	57	Author name and address
1510-1690	372	"HELP" Command
2005-2720	856	"FIND" & "OUTLINE" Commands

## OTHER APPLICATIONS

### 1.) Structured Design (Including Warnier-Orr Diagrams\*):

```
-----  
PAYROLL PROGRAM  
  COMPUTE STATE TAX SUBROUTINE  
    IF STATE = "OHIO"  
-----
```

```
1)+ THEN  
2)+ ELSE  
COMMAND-----)
```

\*See "Structured Programming with Warnier-Orr Diagrams" by D.A. Higgins  
in the December 1977 BYTE Magazine.

### 2.) Planning

```
-----  
BUSINESS PLAN  
1978  
-----
```

```
1)+ GOALS  
2)+ OBJECTIVES  
3)+ CRITICAL PROBLEMS  
4)+ FINANCIAL REPORT  
5)+ CURRENT PROJECTS  
COMMAND-----)
```



### 3.) Family Tree

-----  
DOE FAMILY  
JOHN DOE 9/5/1942  
-----

- 1.) MOTHER: MARY SUE (SMITH) 5/20/1920
- 2.) FATHER: BILL DOE 3/14/1918
- 3.) WIFE: MARY JO (BROWN) 4/17/1944
- 4.)+ CHILDREN
- 5.)+ BIRTHPLACE

### 4.) Ham Radio Directory

-----  
HAM DIRECTORY  
USA  
K8XXXXXX  
-----

- 1.) NAME: BILL JOHNSON
- 2.)+ ADDRESS
- 3.) WIFE: BETTY
- 4.)+ CHILDREN
- 5.)+ BASE: HEATHKIT
- 6.)+ MOBILE: HEATHKIT
- 7.)+ LOG

## SUBROUTINES

400     "ADD"/"CHANGE" COMMAND  
420     ADD FUNCTION FOR INSERT COMMAND  
1050    "SELECT" COMMAND  
1200    "INSERT" COMMAND  
1300    "DELETE" COMMAND  
1500    "HELP" COMMAND  
1700    VALIDATE INSERT PATH  
2000    "FIND" COMMAND  
2400    HIERARCHY SEARCH/DISPLAY OUTLINE  
7000    DISPLAY DATA  
8000    FIND EMPTY SLOT IN ARRAY  
8100    COMMAND INPUT  
8500    "LOWER" COMMAND  
8600    "HIGHER" COMMAND  
8700    PRINT PATH  
8800    "TOP" COMMAND

ROUTINES YOU MAY WANT TO MODIFY

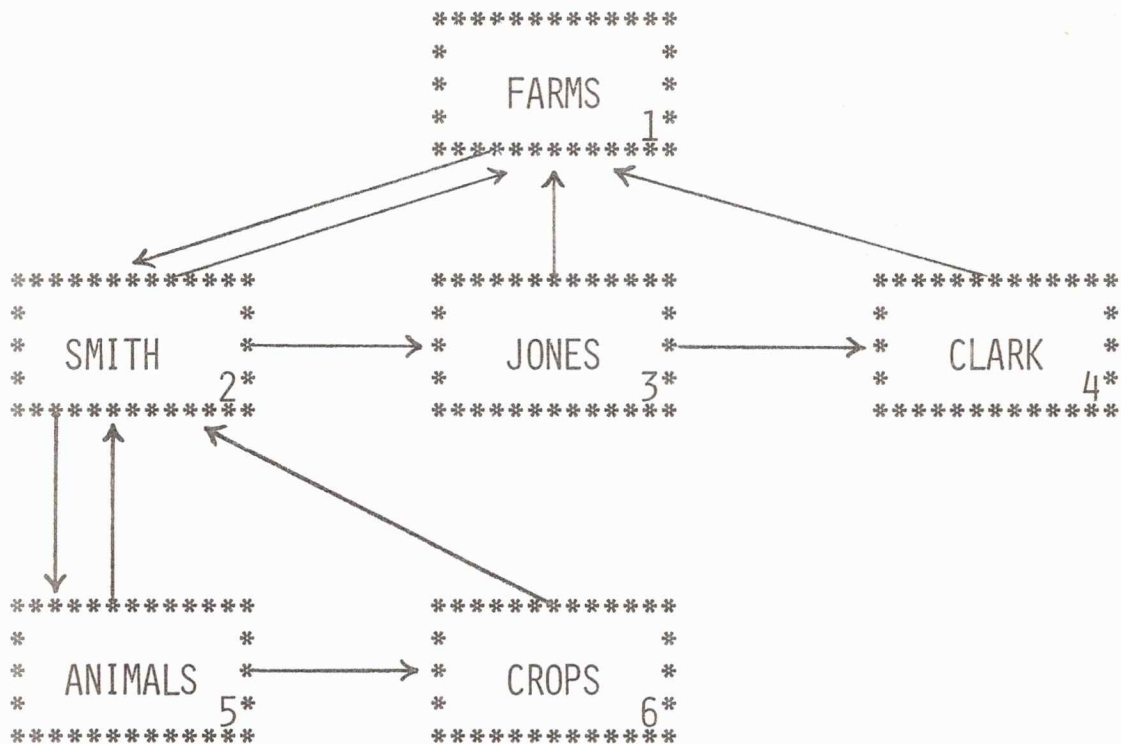
<u>STATEMENT #</u>	<u>DESCRIPTION</u>
75-76	<u>FOR HARDCOPY CONSOLE</u> SET B\$ = CHR\$(10)
8280	<u>FOR HARDCOPY CONSOLE</u> GOSUB 7000 CAUSES A REDISPLAY AFTER EACH COMMAND. YOU MAY WANT TO REMOVE IT.
120	<u>DATA BASE SIZE</u> A9 CONTROLS THE MAXIMUM NUMBER OF DATA ITEMS THAT THE DATA BASE CAN CONTAIN. DEFAULT IS 55 FOR A 16K SYSTEM. ADD ABOUT 20 OR 30 FOR EACH ADDITIONAL 1K OVER THE 16K BASE.
105	<u>DATA BASE LEVELS</u> L9 CONTROLS THE MAXIMUM NUMBER OF LEVELS THE DATA BASE CAN CONTAIN.
1468	<u>PAUSE BETWEEN MULTIPLE DELETES</u> DEFAULT PAUSE 800.
1465	<u>DISABLE MULTIPLE DELETE</u> CHANGE TO: GOTO 1499
1510-1690	<u>ADD ADDITIONAL "HELP" INFORMATION</u>
2250 & 2270	<u>CLOCK CONTROL</u> CLOCK IS STOPPED TO HELP PERFORMANCE OF "FIND" COMMAND.
2521	<u>"OUTLINE" INDENTATION</u> INDENTATION IS CONTROLLED BY THE TAB(L5*2). CHANGE THE "2" TO THE INDENTATION AMOUNT DESIRED.
2525	<u>BELL DURING "FIND"</u> DELETE STATEMENT IF THE BELL IS NOT DESIRED.
7035	<u>PRINT LINE PAUSE ON LARGE DISPLAY</u> IF MORE THAN 11 LINES ARE DISPLAYED A PAUSE IS ISSUED BETWEEN PRINT LINES.

## VARIABLES

A(A9,3)	ARRAY OF DATA BASE POINTERS
A\$(A9)	ARRAY OF DATA BASE DATA
A9	DIM SIZE OF A() AND A\$()
B\$	STRING OF NEW LINE CHARACTERS USED TO CLEAR SCREEN
C\$	ALPHA CHARACTER OF COMMAND
C2	SELECTED ITEM'S DISPLACEMENT IN ARRAY
C9\$	LAST COMMAND ENTERED
D\$	DATA FOR ADD OR CHANGE
D1	EQUATE FOR DAUGHTER POINTER
L	CURRENT LEVEL WITHIN HIERARCHY
L(L9)	CURRENT PATH WITHIN HIERARCHY
L9	MAXIMUM NUMBER OF LEVELS ALLOWED
M	MOTHER DATA ITEM FOR CURRENT POSITION
M1	EQUATE FOR MOTHER POINTER
N	ARRAY INDEX NUMBER RETURNED FROM "FIND SLOT"
N2	NEXT SLOT "FIND SLOT" WILL CHECK FOR ASSIGNMENT
P1	NUMERIC PORTION OF COMMAND
S\$	STRING DATA TO BE FOUND ON "FIND"
S1	EQUATE FOR SISTER POINTER
S2	ITEM PRECEEDING SELECTED ITEM
T1	NUMBER OF ITEMS TO DELETE ON MULTIPLE DELETE
T2	ITEM NUMBER FOR MULTIPLE DELETE



## INTERNAL STRUCTURE



<u>M1</u>	<u>D1</u>	<u>S1</u>	
A(1) = 0, 2, 0			A\$(1) = "FARMS"
A(2) = 1, 5, 3			A\$(2) = "SMITH"
A(3) = 1, 16, 4			A\$(3) = "JONES"
A(4) = 1, 29, 0			A\$(4) = "CLARK"
A(5) = 2, 7, 6			A\$(5) = "ANIMALS"
A(6) = 2, 14, 0			A\$(6) = "CROPS"
A(7) = 5, 0, 8			A\$(7) = "PIGS"
A(8) = 5, 0, 9			A\$(8) = "SHEEP"
A(9) = 5, 11, 10			A\$(9) = "COWS"
	,		,
	,		,
	,		,

M1 IS THE DATA ITEM'S MOTHER.  
 D1 IS THE DATA ITEM'S DAUGHTER.  
 S1 IS THE DATA ITEM'S SISTER.

CHANGING THE SIZE OF AN EXISTING DATA BASE

```
100 INPUT "ENTER NEW VALUE FOR A9. A9 = ";A8
110 DIM Z$(MIN(A8,A9)), Z(MIN(A8,A9),3)
120 FOR I=0 TO MIN(A8,A9)
130 Z$(I) = A$(I)
140 A$(I) = ""
150 Z(I,M1) = A(I,M1)
160 Z(I,D1) = A(I,D1)
170 Z(I,S1) = A(I,S1)
180 PRINT I
190 NEXT I
200 CLEAR A$(
210 CLEAR A(
220 DIM A$(A8), A(A8,3)
240 FOR I = 0 TO MIN(A8,A9)
250 A$(I) = Z$(I)
260 Z$(I) = ""
270 A(I,M1) = Z(I,M1)
280 A(I,D1) = Z(I,D1)
290 A(I,S1) = Z(I,S1)
300 PRINT I
310 NEXT I
320 CLEAR Z$(
330 CLEAR Z(
340 A9 = A8
350 END
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON

```
00010 PRINT :PRINT :PRINT :PRINT :PRINT :PRINT :PRINT :PRINT :PRI
NT :PRINT :PRINT :PRINT :PRINT :PRINT
00020 PRINT "                CREATE A CUSTOM PROGRAM"
00025 PRINT :PRINT "                WRITTEN FOR BENTON HARBOR EXTEN
DED BASIC"
00027 PRINT "                #10.02.01"
00030 PRINT :PRINT "                BY W. C. DENNISON"
00040 PRINT "                7404 STONE BLUFF CT."
00050 PRINT "                LOUISVILLE, KY. 40291"
00060 PRINT "                (502)491-8580"
00070 PRINT :PRINT :PRINT
00080 PRINT "                WITH THIS PROGRAM YOU MAY CUSTOM IT TO DO SEVE
RAL"
00090 PRINT "                DIFFERENT THINGS, SUCH AS : "
00100 GOSUB 1000
00110 PAUSE [3000]
00120 PRINT " YOU WILL BE ABLE TO CUSTOM BUILD THIS PROGRAM TO YO
UR "
00130 PRINT " NEEDS. IT CAN BE USED FOR BUSINESS OR PERSONAL APPL
ICATIONS."
00140 PRINT " THERE ARE MANY VARIABLES AVAILABLE TO YOU:"
00150 PRINT "                A$ = LAST NAME                A1$ = FIRST NAME & INI
TIAL"
00160 PRINT "                S$ = STREET ADDRESS                C$ = CITY"
00170 PRINT "                S1$= STATE                C1$ = ZIP CODE"
00180 PRINT "                P$ = PHONE NUMBER                B$ = INFORMATION"
00190 PRINT "                D$(1 TO 9) OPEN FOR EXPANDED USE"
00200 PAUSE [3000]
00610 PRINT :PRINT :PRINT "                WHAT IS YOUR PLEASURE ? "
00620 GOSUB 1000
00630 PRINT :INPUT "                SELECT BY NUMBER WHAT YOU WANT DONE ? ";
S
00640 ON S GOTO 2000,4100,6000,6500,7000,5000,10000,9000,14250,15
000
01000 PRINT "                1 - MAKE A NEW LIST"
01010 PRINT "                2 - SORT LIST"
01020 PRINT "                3 - PUT LIST ON TAPE"
01030 PRINT "                4 - GET EXISTING LIST FROM TAPE"
01040 PRINT "                5 - ADD TO LIST"
01050 PRINT "                6 - UPDATE LIST"
01060 PRINT "                7 - PRINT LIST BY CATEGORY"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
01070 PRINT "                8 - PRINT ALL INFORMATION"
01080 PRINT "                9 - LOCATE A PARTICULAR ITEM IN LIST
"
01090 PRINT "                10 - PRINT MAILING LABELS"
01100 RETURN
02000 REM      PROGRAM TO MAKE A NEW LIST
02001 IF L1>0 THEN PRINT "      TO MAKE A NEW PROGRAM WILL ERASE TH
E ONE"
02002 IF L1>0 THEN LINE INPUT "      NOW IN MEMORY. ARE YOU SURE ? (Y
OR N) ";M$
02003 IF L1>0 THEN IF M$<>"Y" GOTO 610
02005 L1=0
02010 CLEAR
02020 PRINT "                PROGRAM TO MAKE A NEW LIST"
02030 PRINT
02040 LINE INPUT "      FILE NAME ? ";Z$
02050 PRINT
02060 LINE INPUT "      DO YOU WISH: LAST NAME ? ";L$
02070 IF L$<>"Y" GOTO 2090
02075 GOSUB 2800
02080 DIM A$(L1)
02090 LINE INPUT "      DO YOU WISH: FIRST NAME & INITIAL (Y OR N)
? ";F$
02100 IF F$<>"Y" GOTO 2120
02105 GOSUB 2800
02110 DIM A1$(L1)
02120 LINE INPUT "      DO YOU WISH: STREET ADDRESS ? ";S4$
02130 IF S4$<>"Y" GOTO 2150
02135 GOSUB 2800
02140 DIM S$(L1)
02150 LINE INPUT "      DO YOU WISH: CITY ? ";C4$
02160 IF C4$<>"Y" GOTO 2180
02165 GOSUB 2800
02170 DIM C$(L1)
02180 LINE INPUT "      DO YOU WISH: STATE ? ";T4$
02190 IF T4$<>"Y" GOTO 2210
02195 GOSUB 2800
02200 DIM S1$(L1)
02210 LINE INPUT "      DO YOU WISH: ZIP CODE ? ";Z4$
02220 IF Z4$<>"Y" GOTO 2240
02225 GOSUB 2800
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
02230 DIM C1$(L1)
02240 LINE INPUT "      DO YOU WISH: PHONE NUMBER ? ";P4$
02250 IF P4$<>"Y" GOTO 2270
02255 GOSUB 2800
02260 DIM P$(L1)
02270 LINE INPUT "      DO YOU WISH: INFORMATION ? ";B4$
02280 IF B4$<>"Y" GOTO 2300
02285 GOSUB 2800
02290 DIM B$(L1)
02300 LINE INPUT "      DO YOU WISH: MORE CATEGORIES ? ";M4$
02305 GOSUB 2900
02310 IF M4$<>"Y" GOTO 3000
02315 PRINT :PRINT "      WHEN YOU ARE THRU TYPE <END>."
02320 LINE INPUT "      CATEGORY #1 TO BE TITLED ? ";D$
02330 IF D$="END" GOTO 3000
02340 GOSUB 2800
02350 DIM D$(L1)
02360 LINE INPUT "      CATEGORY #2 TO BE TITLED ? ";D1$
02370 IF D1$="END" GOTO 3000
02380 GOSUB 2800
02390 DIM D1$(L1)
02400 LINE INPUT "      CATEGORY #3 TO BE TITLED ? ";D2$
02410 IF D2$="END" GOTO 3000
02420 GOSUB 2800
02430 DIM D2$(L1)
02440 LINE INPUT "      CATEGORY #4 TO BE TITLED ? ";D3$
02450 IF D3$="END" GOTO 3000
02460 DIM D3$(L1)
02470 LINE INPUT "      CATEGORY #5 TO BE TITLED ? ";D4$
02480 IF D4$="END" GOTO 3000
02490 DIM D4$(L1)
02500 LINE INPUT "      CATEGORY #6 TO BE TITLED ? ";D5$
02510 IF D5$="END" GOTO 3000
02520 DIM D5$(L1)
02530 LINE INPUT "      CATEGORY #7 TO BE TITLED ? ";D6$
02540 IF D6$="END" GOTO 3000
02550 DIM D6$(L1)
02560 LINE INPUT "      CATEGORY #8 TO BE TITLED ? ";D7$
02570 IF D7$="END" GOTO 3000
02580 DIM D7$(L1)
02590 LINE INPUT "      CATEGORY #9 TO BE TITLED ? ";D8$
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
02592 IF D8$="END" GOTO 3000
02710 DIM D8$(L1)
02720 LINE INPUT "          CATEGORY #10 TO BE TITLED ? ";D9$
02730 IF D9$="END" GOTO 3000
02740 DIM D9$(L1)
02750 GOTO 3000
02800 IF L1=0 GOTO 2950
02810 RETURN
02950 INPUT "    HOW MANY IN THE LIST (100,ETC.) ? ";L1
02960 RETURN
02980 D$="END":D1$="END":D2$="END":D3$="END":D4$="END":D5$="END"
02990 D6$="END":D7$="END":D8$="END":D9$="END":C2$="END"
02993 C2$="END"
02995 RETURN
03000 REM  MAKE NEW LIST
03010 V=0
03020 PRINT "    YOUR PROGRAM WILL CONTAIN THE FOLLOWING VARIABLES
:"
03025 GOSUB 64000
03300 PRINT "    YOUR PROGRAM HAS ";V;" VARIABLES. EACH ONE CAN"
03310 PRINT "    CAN CONTAIN ";L1;" ITEMS. EACH CATEGORY WILL BE"
03320 PRINT "    PRESENTED BY PROMPTS AND YOU ARE TO FILL IN THE"
03330 PRINT "    INFORMATION. WHEN YOU ARE THROUGH ENTER - END."
03335 N1=1
03338 N2=N1
03340 FOR I=N2 TO L1:N2=N2+1
03345 PRINT :PRINT
03350 IF L$<>"Y" GOTO 3370
03360 LINE INPUT "          LAST NAME: ";A$(I)
03365 IF A$(I)="END" GOTO 3900
03370 IF F$<>"Y" GOTO 3390
03380 LINE INPUT "          FIRST NAME & INITIAL: ";A1$(I)
03385 IF A1$(I)="END" GOTO 3900
03390 IF S4$<>"Y" GOTO 3410
03400 LINE INPUT "          STREET ADDRESS: ";S$(I)
03405 IF S$(I)="END" GOTO 3900
03410 IF C4$<>"Y" GOTO 3430
03420 LINE INPUT "          CITY: ";C$(I)
03425 IF C$(I)="END" GOTO 3900
03430 IF T4$<>"Y" GOTO 3450
03440 LINE INPUT "          STATE: ";S1$(I)
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
03445 IF S1$(I)="END" GOTO 3900
03450 IF Z4$(<)"Y" GOTO 3470
03460 LINE INPUT "      ZIP CODE: ";C1$(I)
03465 IF C1$(I)="END" GOTO 3900
03470 IF P4$(<)"Y" GOTO 3490
03480 LINE INPUT "      PHONE NUMBER: ";P$(I)
03485 IF P$(I)="END" GOTO 3900
03490 IF B4$(<)"Y" GOTO 3510
03500 LINE INPUT "      INFORMATION: ";B$(I)
03505 IF B$(I)="END" GOTO 3900
03510 IF M4$(<)"Y" GOTO 3830
03525 IF D$="END" GOTO 3555
03530 IF D$(<)" " THEN PRINT "      ";D$::LINE INPUT ":";D$(I)
03540 IF D$(I)="END" GOTO 3900
03555 IF D1$="END" GOTO 3585
03560 IF D1$(<)" " THEN PRINT "      ";D1$::LINE INPUT ":";D1$(I)
03570 IF D1$(I)="END" GOTO 3900
03585 IF D2$="END" GOTO 3615
03590 IF D2$(<)" " THEN PRINT "      ";D2$::LINE INPUT ":";D2$(I)
03610 IF D2$(I)="END" GOTO 3900
03615 IF D3$="END" GOTO 3635
03620 IF D3$(<)" " THEN PRINT "      ";D3$::LINE INPUT ":";D3$(I)
03630 IF D3$(I)="END" GOTO 3900
03635 IF D4$="END" GOTO 3675
03640 IF D4$(<)" " THEN PRINT "      ";D4$::LINE INPUT ":";D4$(I)
03650 IF D4$(I)="END" GOTO 3900
03675 IF D5$="END" GOTO 3705
03680 IF D5$(<)" " THEN PRINT "      ";D5$::LINE INPUT ":";D5$(I)
03700 IF D5$(I)="END" GOTO 3900
03705 IF D6$="END" GOTO 3735
03710 IF D6$(<)" " THEN PRINT "      ";D6$::LINE INPUT ":";D6$(I)
03730 IF D6$(I)="END" GOTO 3900
03735 IF D7$="END" GOTO 3765
03740 IF D7$(<)" " THEN PRINT "      ";D7$::LINE INPUT ":";D7$(I)
03750 IF D7$(I)="END" GOTO 3900
03765 IF D8$="END" GOTO 3795
03770 IF D8$(<)" " THEN PRINT "      ";D8$::LINE INPUT ":";D8$(I)
03790 IF D8$(I)="END" GOTO 3900
03795 IF D9$="END" GOTO 3830
03797 IF D9$(I)="END" GOTO 3900
03800 IF D9$(<)" " THEN PRINT "      ";D9$::LINE INPUT ":";D9$(I)
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
03830 IF C2#<>"END" GOTO 5070
03840 NEXT I
03900 N1=N1-1
03901 N1=N1-1:LINE INPUT " ARE YOU THRU ENTERING DATA (Y OR N)
? ";G#
03910 IF G#<>"Y" GOTO 3350
03920 IF G#="Y" GOTO 610
04100 REM SORT PROGRAM
04110 PRINT " YOU WILL BE ABLE TO SORT YOUR PROGRAM BY THE FO
LLOWING:"
04120 GOSUB 64000
04130 LINE INPUT " SORT THIS PROGRAM BY - ";G#
04145 GOSUB 64500
04147 CLEAR X#(:DIM X#(18)
04150 X#(1)=A#:X#(2)=A1#:X#(3)=S#:X#(4)=C#:X#(5)=S1#:X#(6)=C1#:X#
(7)=P#
04160 X#(8)=B#:X#(9)=D#:X#(10)=D1#:X#(11)=D2#:X#(12)=D3#:X#(13)=D
4#:X#(14)=D5#
04170 X#(15)=D6#:X#(16)=D7#:X#(17)=D8#:X#(18)=D9#
04180 FOR I=1 TO 20
04190 IF G#=X#(I) GOTO 4210
04200 NEXT I
00114 PRINT " YOU HAVE CHOSEN TO SORT THIS PROGRAM BY ";X#(
I)
04220 CLEAR H#(:DIM H#(N1+N1)
04225 FOR I=1 TO N1
04230 IF X#(1)=G# THEN H#(I)=A#(I)
04240 IF X#(2)=G# THEN H#(I)=A1#(I)
04250 IF X#(3)=G# THEN H#(I)=S#(I)
04260 IF X#(4)=G# THEN H#(I)=C#(I)
04270 IF X#(5)=G# THEN H#(I)=S1#(I)
04280 IF X#(6)=G# THEN H#(I)=C1#(I)
04290 IF X#(7)=G# THEN H#(I)=P#(I)
04300 IF X#(8)=G# THEN H#(I)=B#(I)
04310 IF X#(9)=G# THEN H#(I)=D#(I)
04320 IF X#(10)=G# THEN H#(I)=D1#(I)
04330 IF X#(11)=G# THEN H#(I)=D2#(I)
04340 IF X#(12)=G# THEN H#(I)=D3#(I)
04350 IF X#(13)=G# THEN H#(I)=D4#(I)
04360 IF X#(14)=G# THEN H#(I)=D5#(I)
04370 IF X#(15)=G# THEN H#(I)=D6#(I)
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
04380 IF X$(16)=G$ THEN H$(I)=D7$(I)
04390 IF X$(17)=G$ THEN H$(I)=D8$(I)
04400 IF X$(18)=G$ THEN H$(I)=D9$(I)
04410 NEXT I
04420 GOTO 8000
04470 IF L$<>"N" THEN T$=A$(I):A$(I)=A$(L):A$(L)=T$
04480 IF F$<>"N" THEN T$=A1$(I):A1$(I)=A1$(L):A1$(L)=T$
04490 IF S4$<>"N" THEN T$=S$(I):S$(I)=S$(L):S$(L)=T$
04500 IF C4$<>"N" THEN T$=C$(I):C$(I)=C$(L):C$(L)=T$
04510 IF T4$<>"N" THEN T$=S1$(I):S1$(I)=S1$(L):S1$(L)=T$
04520 IF Z4$<>"N" THEN T$=C1$(I):C1$(I)=C1$(L):C1$(L)=T$
04530 IF P4$<>"N" THEN T$=P$(I):P$(I)=P$(L):P$(L)=T$
04540 IF B4$<>"N" THEN T$=B$(I):B$(I)=B$(L):B$(L)=T$
04550 IF M4$="N" THEN GOTO 4660
04560 IF D$<>"END" THEN T$=D$(I):D$(I)=D$(L):D$(L)=T$
04570 IF D1$<>"END" THEN T$=D1$(I):D1$(I)=D1$(L):D1$(L)=T$
04580 IF D2$<>"END" THEN T$=D2$(I):D2$(I)=D2$(L):D2$(L)=T$
04590 IF D3$<>"END" THEN T$=D3$(I):D3$(I)=D3$(L):D3$(L)=T$
04600 IF D4$<>"END" THEN T$=D4$(I):D4$(I)=D4$(L):D4$(L)=T$
04610 IF D5$<>"END" THEN T$=D5$(I):D5$(I)=D5$(L):D5$(L)=T$
04620 IF D6$<>"END" THEN T$=D6$(I):D6$(I)=D6$(L):D6$(L)=T$
04630 IF D7$<>"END" THEN T$=D7$(I):D7$(I)=D7$(L):D7$(L)=T$
04640 IF D8$<>"END" THEN T$=D8$(I):D8$(I)=D8$(L):D8$(L)=T$
04650 IF D9$<>"END" THEN T$=D9$(I):D9$(I)=D9$(L):D9$(L)=T$
04660 RETURN
04980 GOSUB 4470
05000 REM PROGRAM TO UPDATE LIST
05010 PRINT :LINE INPUT " WHAT DO YOU WISH TO CHANGE ? ";C2$
05020 I$=C2$
05030 GOSUB 14020
05040 PRINT " CHANGE THE FOLLOWING OR RETYPE THE INFORMATION .
"
05050 N1=N2:PRINT
05060 N2=I:GOTO 3340
05070 PRINT :LINE INPUT " DO YOU WISH TO CHANGE MORE (Y OR N) ?
";M4$
05075 GOSUB 2993
05080 IF M4$<>"N" GOTO 5010
05090 GOTO 610
06000 REM PROGRAM TO PUT LIST ON TAPE
06010 PRINT :PRINT :PRINT :PRINT
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
06020 PRINT "      TO PUT LIST ON TAPE, SET UP CASSETTE RECORDER"
06030 PRINT " TO ACCEPT INFORMATION. WHEN READY HIT SPACE BAR."
06040 PAUSE
06050 PRINT " NAME OF FILE TO BE STORED ";Z$
06070 GOSUB 65100
06080 STOP
06090 GOTO 610
06500 REM PROGRAM TO GET FILE FROM TAPE
06510 PRINT "
08224 READY TAPE WITH FILE ON IT IN RECORDER. "
06520 PRINT :PRINT " PRESS SPACE BAR WHEN READY"
06545 CLEAR Z$
06550 LINE INPUT " FILE NAME WANTED ? ";Z$
06560 GOSUB 65000
06570 STOP
06580 GOTO 610
07000 REM ADD TO LIST
07010 PRINT " YOU HAVE SELECTED TO ADD TO YOUR LIST. "
07020 PRINT " JUST ANSWER THE PROMPTS AND TYPE 'END'"
07030 PRINT " WHEN YOU ARE THRU."
07040 I=N1
07050 GOTO 3340
07990 I=N1
08000 N=I-1:I=0
08030 REM
08040 M=N
08050 M=INT(M/2)
08060 IF M=0 THEN 8180
08065 M=M-1
08070 J=1:K=N-M
08080 I=J
08090 L=I+M
08100 IF H$(I)<H$(L) GOTO 8150
08110 GOSUB 4470
08120 I=I-M
08130 IF I<1 THEN 8150
08140 GOTO 8090
08150 J=J+1
08160 IF J>K THEN 8060
08170 GOTO 8030
08180 PRINT " SORT COMPLETE !!!"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
08190 GOSUB 63000
08200 GOTO 610
09000 REM PRINT PROGRAM
09010 FOR I=1 TO N2
09020 GOSUB 35000
09200 PRINT :PRINT :NEXT I
09210 PRINT "          PRINT OUT OF ALL INFORMATION IS COMPLETE. "
09220 PRINT "          TO CONTINUE PRESS SPACE BAR."
09230 PAUSE
09240 GOTO 610
10000 PRINT :PRINT :REM PROGRAM TO PRINT BY CATEGORY
10005 CLEAR M$:CLEAR M$(:DIM M$(18)
10010 PRINT " YOU MAY SELECT FROM THE FOLLOWING :"
10020 GOSUB 64500
10025 GOSUB 64000
10030 PRINT :PRINT " CHOOSE FROM THE LIST WHAT YOU WOULD LIKE T
0 "
10040 N=0
10050 PRINT " HAVE PRINTED. "
10060 PRINT :PRINT " ENTER 'END' WHEN YOU ARE THRU ."
10070 N=N+1
10075 I=N
10080 PRINT :LINE INPUT " PRINT BY - ";M$(I)
10085 IF N=18 GOTO 10100
10090 IF M$(N)<>"END" GOTO 10070
10095 N=N-1
10100 FOR B=1 TO N2
10110 FOR I=1 TO N
10120 IF M$(I)=A$THEN PRINT TAB(10);A$(B);", ";
10130 IF M$(I)=A1$THEN PRINT A1$(B)
10140 IF M$(I)=S$THEN PRINT TAB(10);S$(B)
10150 IF M$(I)=C$THEN PRINT TAB(10);C$(B);", ";
10160 IF M$(I)=S1$THEN PRINT S1$(B);" ";
10170 IF M$(I)=C1$THEN PRINT C1$(B)
10180 IF M$(I)=P$THEN PRINT :PRINT TAB(15);P$;"- ";P$(B)
10190 IF M$(I)=B$THEN PRINT :PRINT TAB(5);B$;"- ";B$(B)
10200 IF M$(I)=D$THEN PRINT D$;"- ";D$(B)
10210 IF M$(I)=D1$THEN PRINT D1$;"- ";D1$(B)
10220 IF M$(I)=D2$THEN PRINT D2$;"- ";D2$(B)
10230 IF M$(I)=D3$THEN PRINT D3$;"- ";D3$(B)
10240 IF M$(I)=D4$THEN PRINT D4$;"- ";D4$(B)
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
10250 IF M$(I)=D5$ THEN PRINT D5$;"- ";D5$(B)
10260 IF M$(I)=D6$ THEN PRINT D6$;"- ";D6$(B)
10270 IF M$(I)=D7$ THEN PRINT D7$;"- ";D7$(B)
10280 IF M$(I)=D8$ THEN PRINT D8$;"- ";D8$(B)
10290 IF M$(I)=D9$ THEN PRINT D9$;"- ";D9$(B)
10300 PRINT :NEXT I
10310 NEXT B
10320 FOR I=1 TO N
10600 PRINT " PRINT OUT IS COMPLETE. TO CONTINUE PRESS"
10610 PRINT " SPACE BAR. "
10620 PAUSE
10630 GOTO 610
14000 REM LOCATE A PARTICULAR ITEM IN LIST
14005 PRINT :PRINT
14010 LINE INPUT " WHAT DO YOU WISH TO FIND ? ";I$
14020 FOR I=1 TO N2
14030 IF L$<>"N" THEN IF I$=A$(I) THEN GOTO 14500
14040 IF F$<>"N" THEN IF I$=A1$(I) THEN GOTO 14500
14050 IF S4$<>"N" THEN IF I$=S$(I) THEN GOTO 14500
14060 IF C4$<>"N" THEN IF I$=C$(I) THEN GOTO 14500
14070 IF T4$<>"N" THEN IF I$=S1$(I) THEN GOTO 14500
14080 IF Z4$<>"N" THEN IF I$=C1$(I) THEN GOTO 14500
14090 IF P4$<>"N" THEN IF I$=P$(I) THEN GOTO 14500
14100 IF B4$<>"N" THEN IF I$=B$(I) THEN GOTO 14500
14110 IF M4$="N" GOTO 14230
14120 IF D$<>"END" THEN IF I$=D$(I) GOTO 14500
14130 IF D1$<>"END" THEN IF I$=D1$(I) GOTO 14500
14140 IF D2$<>"END" THEN IF I$=D2$(I) GOTO 14500
14150 IF D3$<>"END" THEN IF I$=D3$(I) GOTO 14500
14160 IF D4$<>"END" THEN IF I$=D4$(I) GOTO 14500
14170 IF D5$<>"END" THEN IF I$=D5$(I) GOTO 14500
14180 IF D6$<>"END" THEN IF I$=D6$(I) GOTO 14500
14200 IF D7$<>"END" THEN IF I$=D7$(I) GOTO 14500
14210 IF D8$<>"END" THEN IF I$=D8$(I) GOTO 14500
14220 IF D9$<>"END" THEN IF I$=D9$(I) GOTO 14500
14230 NEXT I
14240 RETURN
14250 REM SEARCH ROUTINE
14260 PRINT :PRINT :LINE INPUT " WHAT DO YOU WISH TO FIND ? ";I$
14270 GOSUB 14020
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
14280 PRINT :PRINT I$;" NOT FOUND IN LIST. "
14290 PRINT :GOSUB 35000
14300 PRINT :GOSUB 63000
14310 GOTO 610
14500 PRINT I$;" FOUND AS THE #";I;" IN THE LIST. "
14510 PRINT :GOSUB 35000
14520 IF C2$<>"END" THEN RETURN
14525 GOSUB 63000
14540 GOTO 610
15000 REM PRINT MAILING LABEL PROGRAM
15010 PRINT "      THIS PROGRAM WILL PRINT THE FOLLOWING:"
15020 PRINT "      FIRST NAME      MIDDLE INITIAL      LAST NAME
"
15030 PRINT "      STREET ADDRESS"
15040 PRINT "      CITY, STATE      ZIP CODE"
15045 GOSUB 15500
15050 PRINT :PRINT "      TO CHANGE THIS GOTO 'LINES 15000 TO 15200'
"
15060 PRINT "      IN THIS PROGRAM AND CHANGE THE VARIABLES FOR TH
E"
15070 PRINT "      ONES THAT YOU WANT ! "
15080 PRINT :PRINT "      READY LABELS IN PRINTING DEVICE AND PRESS
SPACE BAR"
15090 PRINT "      WHEN READY. "
15100 PAUSE
15110 PRINT " MAILING LABELS FOR"
15120 PRINT " LIST ";Z$
15130 PRINT :PRINT :PRINT :PRINT
15140 FOR I=1 TO N2
15150 PRINT A1$(I);A$(I)
15160 PRINT S$(I)
15170 PRINT C$(I);", ";S1$(I);"      ";C1$(I)
15180 PRINT :PRINT :PRINT
15200 NEXT I
15500 IF L$<>"Y" THEN 15600
15510 IF F$<>"Y" THEN 15600
15520 IF S4$<>"Y" THEN 15600
15530 IF C4$<>"Y" THEN 15600
15540 IF T4$<>"Y" THEN 15600
15600 PRINT :PRINT "      ALL THE ABOVE INFORMATION IS NOT AVAILABL
E. "
15700 RETURN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
35000 REM PRINT ROUTINE
35010 IF L$<>"N" THEN PRINT A$(I);
35020 IF F$<>"N" THEN PRINT ", ";A1$(I)
35030 IF S4$<>"N" THEN PRINT TAB(5);S$(I)
35040 IF C4$<>"N" THEN PRINT TAB(5);C$(I);", ";
35050 IF T4$<>"N" THEN PRINT S1$(I);" ";
35060 IF Z4$<>"N" THEN PRINT C1$(I)
35070 IF P4$<>"N" THEN PRINT " PHONE # ";P$(I)
35080 IF B4$<>"N" THEN PRINT " INFORMATION: ";B$(I)
35090 IF D$<>"END" THEN PRINT " ";D$;"- ";D$(I)
35100 IF D1$<>"END" THEN PRINT " ";D1$;"- ";D1$(I)
35110 IF D2$<>"END" THEN PRINT " ";D2$;"- ";D2$(I)
35120 IF D3$<>"END" THEN PRINT " ";D3$;"- ";D3$(I)
35130 IF D4$<>"END" THEN PRINT " ";D4$;"- ";D4$(I)
35140 IF D5$<>"END" THEN PRINT " ";D5$;"- ";D5$(I)
35150 IF D6$<>"END" THEN PRINT " ";D6$;"- ";D6$(I)
35160 IF D7$<>"END" THEN PRINT " ";D7$;"- ";D7$(I)
35170 IF D8$<>"END" THEN PRINT " ";D8$;"- ";D8$(I)
35180 IF D9$<>"END" THEN PRINT " ";D9$;"- ";D9$(I)
35200 RETURN
60080 U=U+1:PRINT TAB(30-LEN(S$));": "
63000 PRINT :PRINT :PRINT " TO CONTINUE PRESS SPACE BAR. "
63010 PAUSE
63020 RETURN
64000 REM GET VARIABLES SET UP IN CUSTOM PROGRAM
64005 GOSUB 64500
64010 IF L$<>"Y" GOTO 64050
64040 U=U+1:PRINT TAB(30-LEN(A$));A$;": "
64050 IF F$<>"Y" GOTO 64070
64060 U=U+1:PRINT TAB(30-LEN(A1$));A1$;": "
64070 IF S4$<>"Y" GOTO 64090
64080 U=U+1:PRINT TAB(30-LEN(S$));S$;": "
64090 IF C4$<>"Y" GOTO 64110
64100 U=U+1:PRINT TAB(30-LEN(C$));C$;": "
64110 IF T4$<>"Y" GOTO 64130
64120 U=U+1:PRINT TAB(30-LEN(S1$));S1$;": "
64130 IF Z4$<>"Y" GOTO 64150
64140 U=U+1:PRINT TAB(30-LEN(C1$));C1$;": "
64150 IF P4$<>"Y" GOTO 64170
64160 U=U+1:PRINT TAB(30-LEN(P$));P$;": "
64170 IF B4$<>"Y" GOTO 64190
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CREATE A PROGRAM - DENNISON < CONT'D >

```
64180 U=U+1:PRINT TAB(30-LEN(B$));B$;" "
64190 IF M4$<>"Y" GOTO 64500
64200 IF D$="END" GOTO 64500
64210 U=U+1:PRINT TAB(30-LEN(D$));D$;" "
64220 IF D1$="END" GOTO 64500
64230 U=U+1:PRINT TAB(30-LEN(D1$));D1$;" "
64240 IF D2$="END" GOTO 64500
64250 U=U+1:PRINT TAB(30-LEN(D2$));D2$;" "
64260 IF D3$="END" GOTO 64500
64270 U=U+1:PRINT TAB(30-LEN(D3$));D3$;" "
64280 IF D4$="END" GOTO 64500
64290 U=U+1:PRINT TAB(30-LEN(D4$));D4$;" "
64300 IF D5$="END" GOTO 64500
64310 U=U+1:PRINT TAB(30-LEN(D5$));D5$;" "
64320 IF D6$="END" GOTO 64500
64330 U=U+1:PRINT TAB(30-LEN(D6$));D6$;" "
64340 IF D7$="END" GOTO 64500
64350 U=U+1:PRINT TAB(30-LEN(D7$));D7$;" "
64360 IF D8$="END" GOTO 64500
64370 U=U+1:PRINT TAB(30-LEN(D8$));D8$;" "
64380 IF D9$="END" GOTO 64500
64390 U=U+1:PRINT TAB(30-LEN(D9$));D9$;" "
64400 RETURN
64500 A$="LAST NAME":A1$="FIRST NAME $ INITIAL":S$="STREET":C$="C
ITY"
64510 S1$="STATE":C1$="ZIP CODE":P$="PHONE NUMBER":B$="INFORMATIO
N"
64520 RETURN
65000 REM TAPE 'GET' ROUTINE
65010 POKE 8302,85:POKE 8303,78:POKE 8304,13
65020 POKE 8305,71:POKE 8306,69:POKE 8307,90:POKE 8308,36:POKE 83
09,13
65030 POKE 8310,89
65040 POKE 8311,67:POKE 8312,79:POKE 8313,78:POKE 8314,13
65050 POKE 8301,13
65060 RETURN
65100 REM TAPE 'PUT' ROUTINE
65110 POKE 8302,80:POKE 8303,85:POKE 8304,90:POKE 8305,36:POKE 83
06,13
65120 POKE 8307,67:POKE 8308,79:POKE 8309,78:POKE 8310,13
65130 POKE 8301,9
65140 RETURN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: I/O ROUTINES - MASUDA

```
00001 GOSUB 65000
00010 PRINT "1.OPEN FILE AS OUTPUT  7. PUT STRING"
00020 PRINT "2.OPEN FILE AS INPUT   8. WRITE DATA RECORD"
00030 PRINT "3.OPEN FILE AS EXTEND  9. GET NEXT NUMBER"
00040 PRINT "4.CLOSE INPUT FILE    10. GET NEXT STRING"
00050 PRINT "5.CLOSE OUTPUT FILE   11. READ DATA RECORD"
00060 PRINT "6.PUT NUMBER"
00065 PRINT
00070 INPUT "ENTER THE NUMBER OF THE FUNCTION OR -1 TO STOP ";C
00075 IF C=-1 THEN STOP
00080 ON C GOTO 2000,2020,2030,2040,2050,2060,2070,2080,2090,2100
,2110
00090 PRINT "INVALID CHOICE":GOTO 10
02000 REM ***** OPEN FILE AS OUTPUT
02002 PRINT "READY OUTPUT DRIVE AND HIT [RET] TO CONTINUE":PAUSE
02004 LINE INPUT "ENTER OUTPUT LABEL ";Y$
02006 GOSUB 64100
02008 PRINT "DONE" :GOTO 10
02020 REM ***** OPEN FILE AS INPUT
02022 PRINT "READY INPUT DRIVE AND HIT [RET] TO CONTINUE":PAUSE
02024 LINE INPUT "ENTER INPUT LABEL ";X$
02026 GOSUB 64130
02028 GOTO 2008
02030 REM ***** OPEN FILE AS EXTENDED
02032 PRINT "READY INPUT AND OUTPUT DRIVES AND HIT [RET] TO CONTI
NUE":PAUSE
02034 LINE INPUT "ENTER INPUT FILE LABEL ";Y$
02036 GOSUB 64190
02038 GOTO 2008
02040 REM ***** CLOSE INPUT FILE
02042 PRINT "INPUT FILE IS NOW CLOSED"
02044 GOSUB 64230
02046 GOTO 2008
02050 REM ***** CLOSE OUTPUT FILE
02052 PRINT "OUTPUT FILE IS NOW BEING CLOSED"
02054 GOSUB 64240
02056 GOTO 2008
02060 REM ***** PUT NUMBER
02062 INPUT "ENTER A NUMBER OR -1 TO STOP ";Y
02064 IF Y=-1 GOTO 2008
02066 GOSUB 64260
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: I/O ROUTINES - MASUDA < CONT'D >

```
02068 GOTO 2062
02070 REM ***** PUT STRING
02072 LINE INPUT "ENTER A STRING OR 'END' TO STOP ";V$
02074 IF V$="END" GOTO 2008
02076 GOSUB 64290
02078 GOTO 2072
02080 REM ***** WRITE DATA RECORD
02082 PRINT "RECORD BEING WRITTEN"
02084 GOSUB 64310
02086 GOTO 2008
02090 REM ***** GET NUMBER
02092 GOSUB 64360
02094 PRINT "NUMBER RETRIEVED =";X
02096 GOTO 2008
02100 REM ***** GET A STRING
02102 GOSUB 64390
02104 PRINT "STRING RETRIEVED =";X$
02106 GOTO 2008
02110 REM ***** READ A REC
02112 PRINT "READING RECORD"
02114 GOSUB 64440
02116 GOTO 2008
05000 GOSUB 65000:X$="TEST-MSTR":GOSUB 64130
05005 GOSUB 64440
05010 IF PEEK(28680)=2 THEN PRINT "END OF FILE READ":STOP
05015 GOSUB 64390:PRINT X$;TAB(50);
05020 GOSUB 64390:PRINT X$;TAB(65);
05025 GOSUB 64360:PRINT X
05030 GOTO 5005
64000 REM ***** READ A RECORD
64005 POKE 17990,177 : POKE 17991,1
64010 Z$="" : Z9=USR(0)
64015 FOR Z8=28684 TO 28683 + PEEK(28683)
64020 Z$=Z$+CHR$(PEEK(Z8))
64025 NEXT Z8
64030 RETURN
64040 REM ***** WRITE A RECORD
64045 Z9=LEN(Z$) : POKE 28683,Z9
64050 FOR Z8 =1 TO Z9
64055 POKE 28683+Z8,ASC(MID$(Z$,Z8,1))
64060 NEXT Z8
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: I/O ROUTINES - MASUDA < CONT'D >

```
64065 POKE 17990,252 : POKE 17991,1
64070 POKE 8192,10 : POKE 8193,112
64075 POKE 8212,12 : POKE 8213,113
64080 Z9=USR(0)
64085 RETURN
64100 REM ***** OPEN FILE OUTPUT
64105 IF PEEK(28681)=0 GOTO 64115
64110 PRINT Z1$: STOP
64115 POKE 28681,1 : POKE 28682,0
64120 Z#=Y$:GOSUB 64040:Y9$=""
64125 RETURN
64130 REM ***** OPEN FILE INPUT
64135 IF PEEK(28680)=0 GOTO 64145
64140 PRINT Z2$: STOP
64145 POKE 28680,1
64150 GOSUB 64000
64155 IF PEEK(28682)=0 GOTO 64165
64160 PRINT Z3$: STOP
64165 X9=-1: IF Z#=X$ THEN RETURN
64170 PRINT Z4$:Z$
64175 LINE INPUT "CONTINUE ? Y/N ":X0$
64180 IF X0$="Y" THEN RETURN
64185 STOP
64190 REM ***** OPEN FILE EXTENDED
64195 X#=Y$:GOSUB 64130
64200 GOSUB 64100
64205 GOSUB 64440
64210 IF PEEK(28680)=2 GOTO 64225
64212 Y9$=X9$
64215 GOSUB 64310
64220 GOTO 64205
64225 GOSUB 64230:Y$="":RETURN
64230 REM ***** CLOSE INPUT FILE
64231 IF PEEK(28680)=1 OR PEEK(28680) = 2 GOTO 64235
64232 PRINT X2$:STOP
64235 POKE 28680,0 : RETURN
64240 REM ***** CLOSE OUTPUT FILE
64241 IF PEEK(28681)=1 GOTO 64243
64242 PRINT X2$:STOP
64243 POKE 28682,2
64245 Z#=Z0$:GOSUB 64040
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: I/O ROUTINES - MASUDA < CONT'D >

```
64250 POKE 28681,0:RETURN
64260 REM ***** PUT A NUMBER
64265 Y8$=STR$(Y)
64270 IF LEN(Y8$)+LEN(Y9$)+1 <=255 GOTO 64280
64275 PRINT Z5$:STOP
64280 Y9$=Y9$+CHR$(28)+Y8$
64285 RETURN
64290 REM ***** PUT A STRING
64295 IF LEN(Y9$)+LEN(Y$)+1 <=255 GOTO 64305
64300 PRINT Z5$:STOP
64305 Y9$=Y9$+CHR$(29)+Y$ : RETURN
64310 REM ***** WRITE A DATA RECORD
64315 IF LEN(Y9$)+1<=255 GOTO 64325
64320 PRINT Z5$:STOP
64325 Y9$=Y9$+CHR$(3)
64330 IF PEEK(28681)=1 GOTO 64340
64335 PRINT Z6$:STOP
64340 POKE 28682,1
64345 Z$=Y9$:GOSUB 64040
64350 Y9$="":RETURN
64360 REM ***** GET A NUMBER
64361 IF X9>=1 GOTO 64365
64362 PRINT X3$:STOP
64365 IF MID$(X9$,X9,1)=CHR$(28) GOTO 64375
64370 PRINT Z7$:STOP
64375 X$="":GOSUB 64410
64380 X=VAL(X$):RETURN
64390 REM ***** GET A STRING
64391 IF X9>=1 GOTO 64395
64392 PRINT X3$:STOP
64395 IF MID$(X9$,X9,1)=CHR$(29) GOTO 64405
64400 PRINT Z7$:STOP
64405 X$="":GOSUB 64410:RETURN
64410 REM ***** COLLECT A STRING
64415 X9=X9+1
64420 X8$=MID$(X9$,X9,1)
64425 IF X8$=CHR$(28) OR X8$=CHR$(29) OR X8$=CHR$(3) THEN RETURN
64430 X$=X$+X8$:GOTO 64415
64440 REM ***** READ A DATA RECORD
64445 IF PEEK(28680)=1 GOTO 64451
64450 PRINT X4$:STOP
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

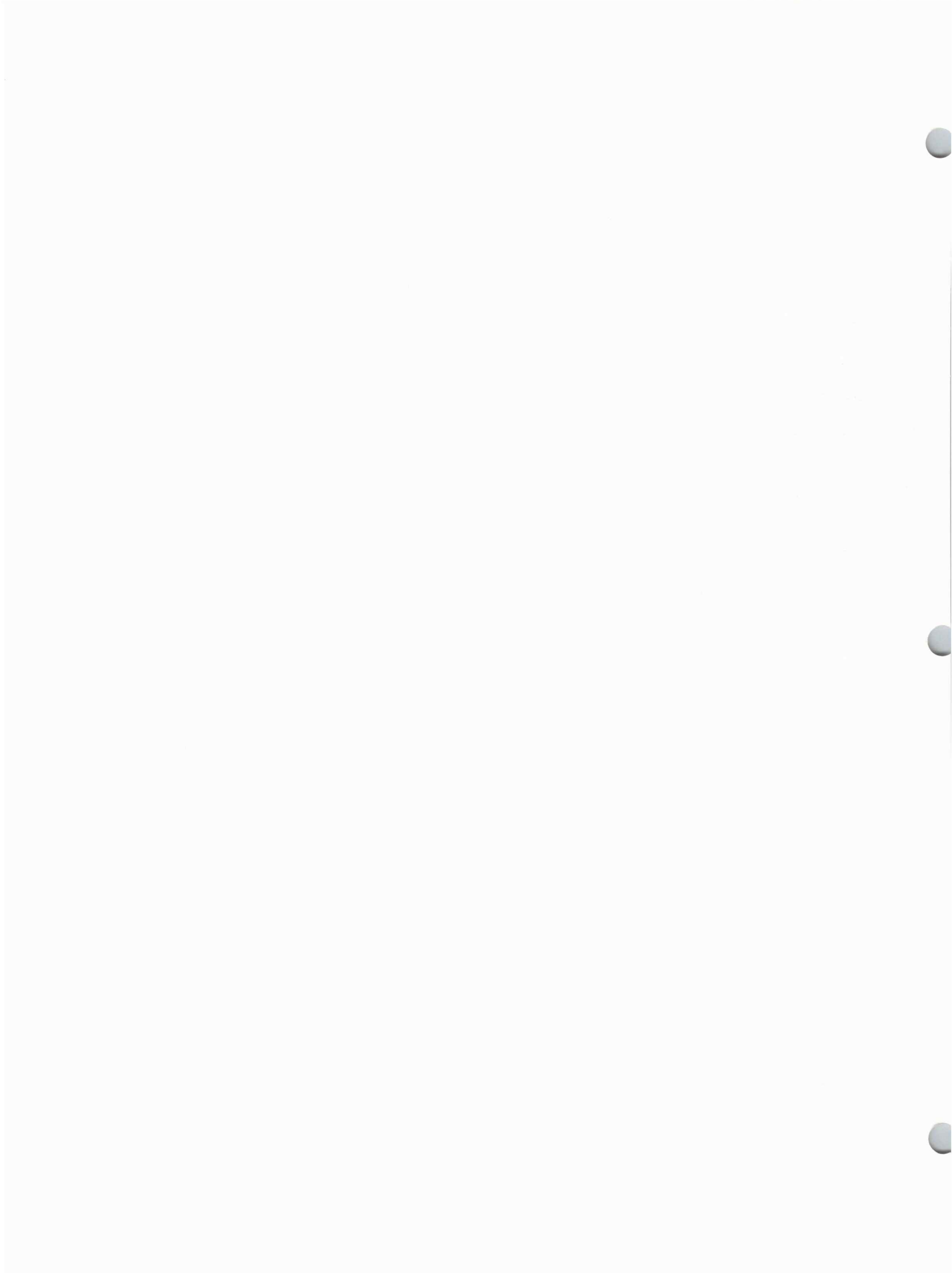
PROGRAM NAME: I/O ROUTINES - MASUDA &lt; CONT'D &gt;

```

64451 IF PEEK(28680)<>2 GOTO 64455
64452 PRINT X4$:STOP
64455 GOSUB 64000
64460 X9$=Z$
64465 IF Z$=Z0$ THEN POKE 28680,2
64470 X9=1:RETURN
65000 REM ***** MESSAGES AND CONSTANTS
65005 Z0$="1EOF-H8"
65010 Z1$="** OUTPUT FILE ALREADY OPENED **"
65015 Z2$="** INPUT FILE ALREADY OPENED **"
65020 Z3$="** HEADER LABEL NOT FOUND **"
65025 Z4$="** LABEL ERROR - LABEL FOUND ="
65030 Z5$="** MAXIMUM RECORD LENGTH EXCEEDED **"
65035 Z6$="** ATTEMPTED WRITE TO UNOPENED FILE **"
65040 Z7$="** TYPE CONFLICT EXISTS **"
65045 X1$="** ATTEMPTING TO READ FROM AN UNOPENED FILE **"
65046 X2$="** ATTEMPTING TO CLOSE A FILE THAT WAS NOT OPENED **"
65047 X3$="** ATTEMPTING TO GET A NUMBER/STRING BEFORE A READ **"
65048 X4$="** ATTEMPTING TO READ PAST END OF FILE **"
65055 POKE 28680,0 : POKE 28681,0
65060 RETURN

```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----



FILENAME = I/O ROUTINES - MASUDA

## PROGRAM DESCRIPTION

The purpose of this program is to demonstrate to the user one method of handling tape operations on the H8 computer.

The program is actually broken up into three distinct sections. Each section will be covered in detail.

## PROGRAMMER:

Richard Masuda  
950 Howe Avenue #23  
Sacramento CA 95825  
1-916-925-8185

---

## CONFIGURATION

Computer:	H8
Memory:	24K RAM
I/O Devices:	H9 Terminal 1 Input Tape Unit 1 Output Tape Unit
Language:	Extended BASIC Version 10.02.01 Utilities PAM-8 Tape Load-Dump Routines

---

## INSTALLATION

The tape supplied has two copies each of Extended BASIC Version 10.02.01 and a BASIC file "I/O Routines".

1. Load BASIC with the PAM-8 loader.
2. Press the "GO" key.
3. Type Load "I/O Routines".
4. Type Continue  
The program menu should be displayed.

This program needs at least 260 words above BASIC to work correctly. The BASIC supplied has been configured with 20K of BASIC and the upper 4K for user workspace.

## MEMORY MAP

<u>DECIMAL</u>	<u>SPLIT</u>	
28680	160 010	Input File Status Byte
28681	160 011	Output File Status Byte
28682	160 012	Record Type
28683	160 013	Record Length
28684	160 014	Beginning of Data Block
28940	161 014	End of Data Block
Input Status Byte	0 = Closed,1=Open,2=EOF Read	
Output Status Byte	0 = Closed,1=Open	
Record Type	0 = Header REC,1=Data REC,2=EOF	

---

SECTION I

Lines 10-2116 represent the interactive section of the program. The user has a menu of tape and data functions displayed on the screen and he picks a function he wishes to perform. All functions will be executed the same way a batch program would work except any necessary information is prompted at the terminal.

See the section on function descriptions for a detailed description of each option.

---

On the tape there is a file names "TEST-MSTR". This tape was created using the interactive section of the program.

## CREATING "TEST-MSTR"

Function #1                    Open File as Output  
Enter "TEST-MSTR" for a label (no quotes)

Function #7                    Put String  
Enter "THE VILLAGE" (RET)  
Enter "483-8148" (RET)  
Enter "END"

Function #6                    Put Number  
Enter "2681"  
Enter -1

Function #8                    Write Data Record

Repeat 7, 6, 8 for as many records as you want.

After you have entered all the information, you must close the output file

Function #5

A data tape has been created.

Put the tape into the reader and open Input File (#2).

Use Function #11                Read Data REC

Use Function #10                Two Times (2 Strings)

Use Function #9                Once (1 Number)

Close the file when you are finished reviewing the contents.

---

"TEST-MSTR" FILE CONTENTS

	A/N	A/N	N
#1	The Village	483-8148	2681
#2	Classic Cabinets	922-6029	1229
#3	Fort Sutter Hearing Aid Centers	966-0336	2724
#4	ABC Company	462-8122	1874
#5	The Electric Company	821-4211	1643
#6	Pacific Gas and Electric	926-8102	1401
#7	EOF Record		



## SECTION II

This section will read the "TEST-MSTR" created on the tape and display it on the terminal. The program is included to show how a batch mode program might work.

To Run

Get to the command level

Ready the tape with the tape at 125 according to the tape indicator.

\*GOTO 5000

\*Continue

The program will print a listing of the tape similar to the listing at the end of Section I.

---

TEST-MSTR LISTING PROGRAM

```
5000  *GOSUB 65000: X$="TEST-MSTR":GOSUB 64130
      Initialization Tape Label Open Input File
5005  GOSUB 64440
      Read a Data Rec
5010  If PEEK(28680)=0 then print "END OF FILE":STOP
      Check for EOF
5015  GOSUB 64390 :PRINTX$;TAB(50);
      Get Name and Print It
5020  GOSUB 64390 :PRINTX$;TAB(65);
      Get Telephone Number and Print It
5025  GOSUB 64360 :PRINTX
      Get Number and Print
5030  GOTO 5005
      Repeat Until EOF Read
```

*\*Must be performed before any tape operations!*

---

### SECTION III

Lines 64000-65055 contain the actual routines involved in the tape I/O operations.

Each routine will be described.

Name of Function

Entry\*=What does the programmer pass to the routine.

Exit\*=What does the programmer get from the routine.

User Called:Does the user call this routine=(yes)

Lines:BASIC Source Lines

Entry Point to GOSUB Statement

\*Where the entry is marked *PROGRAMMER IGNORES* the user only has to call the routine. The program takes care of the entry and exit variables.

Read A Record

Entry

Exit : Z\$=RECORD READ

User Called: *NO*

Lines 64000-64030

Write A Record

Entry : Z\$=RECORD TO WRITE

User Called: *NO*

Lines 64040-64090

Open File Output

Entry : Y\$=HEADER TAPE LABEL

Exit :

User Called: *YES*

Lines 64100-64125

Open File Input

Entry : X\$=HEADER TAPE LABEL

Exit :

User Called: *YES*

Lines 64130-64185

Open File Extended

Entry : X\$=INPUT AND OUTPUT TAPE LABEL

Exit:

User Called: *YES*

Lines: 64190-64225

Close Input File

Entry :

Exit :

User Called: *YES*

Lines: 64230-64235

Close Output File

Entry :

Exit :

User called: *YES*

Lines 64240-64250

Put A Number

Entry : Y=VALUE TO BE OUTPUT

Exit :

User Called : *YES*

Lines: 64264-64285

Put A String

Entry : Y\$=STRING TO BE OUTPUT

Exit :

User Called: *YES*

Lines: 64290-64305

Write A Data Record

Entry : Y9\$ *PROGRAMMER IGNORES*

Exit :

User Called: *YES*

Lines: 64310-64350

Get A Number

Entry :

Exit : X CONTAINS VALUE READ

User Called: *YES*

Lines: 64360-64380

Get A String

Entry :

Exit : X\$ CONTAINS STRING READ

User Called: *YES*

Lines: 64390-64405

Collect A String

Entry : X9\$,X9

Exit : X\$

User Called: *NO*

Lines : 64410-64430

## Read A Data Record

Entry :

Exit : X9\$ *PROGRAMMER IGNORES*

User Called :

Lines: 64440-64470

---

COMMAND SUMMARY

	GOSUB	Required Variables
Open File Output	64100	Y\$=LABEL
Open File Input	64130	X\$=LABEL
Open File Extended	64190	X\$=I/O LABEL
Close Input File	64230	
Close Output File	64240	
Put A Number	64260	Y=VALUE WRITTEN
Put A String	64290	Y\$=STRING WRITTEN
Write A Data Record	64310	
Get A Number	64360	X=VALUE READ
Get A String	64390	X\$=STRING READ
Read A Data Record	64440	

---



## TAPE FUNCTION DESCRIPTIONS

### Open File As Output

The tape output file must be opened before the program can write to tape.

The routine expects a file label to be in Y\$. For example, 10 Y\$="TEST-MSTR" would write the string "TEST-MSTR" as the header record for the file. The label may be up to 256 allowable characters long.

If the file is already open upon entering the routine, an error results and the program aborts.

### Open File As Input

The tape input file must be opened before the program can read from tape.

The routine expects a file label to be in X\$. For example, 10 X\$="TEST-MSTR" would tell the program that the first record on the tape should contain "TEST-MSTR".

If the label provided by the user does not match the label on the tape, the program will display a message (continue? Y/N).

If the user puts a "Y", the program will ignore the label error and continue. Otherwise the program will abort.

If the file is already open upon entering the routine, an error results and the program aborts.

### Open File As Extend

This routine will

1. Open the input file.
2. Open the output file.
3. Copy all of the records from the input file to the output file.
4. NOT write the EOF record to the output file.
5. Close the input file.
6. Returns control to the user.

Because of the sequential nature of tapes, this function allows the user to append data to the original file.

The user is responsible for closing the output file.

If a label error occurs, the label provided by the user will be used on the output file.

### Close The Input File

When the user invokes this function, a zero is stored in the input file status byte to indicate that no more data is available from the input tape.

If the file is already closed, an error results and the program aborts.

### Close Output File

When the output file is closed, three (3) things happen:

1. A check is made to make sure the file was not already closed. If the file is already closed, an error occurs and the program aborts.
2. A record containing the string "1EOF-H8" is written to tape.
3. The output file status byte is set to indicate that writes to the output file are no longer valid.

### Put A Number and Put A String

These routines allow the user to format data into the output record. In both cases, a check is made to make sure that a 255 character maximum is maintained.

When 'Put A Number' is called, the number is converted into ASCII characters and appended to the output record string. One (1) character of control information is also appended. (file separator CHAR\$(28))

When 'Put A String' is called, the string and it's control character is appended to the output record string. Control information consists of the group separator CHR\$(29).

The format of the output record will be:

FS/N<sub>1</sub>/FS/N<sub>2</sub>/GS/S<sub>1</sub> -----GS/S<sub>n</sub>/FS/N<sub>n</sub>/ETX

Where FS = file separator character preceeding all numeric fields.

GS = group separator character preceeding all string fields.

ETX = end of text character.

Put number expects a value in Y.

Put string expects a string value in Y\$.

---

### Write A Data Record

After the user has 'PUT' his numbers and strings, he may write a data record. This routine actually writes the data to tape (assuming the output file is open).

---

### Get A Number and Get A String

The opposite of putting a number/string is

'GET'ting a number or string.

Using this routine the program will decode the input record string and make available the next value.

When 'GET' a number is called, the program will return the value in X.

When 'GET' a string is called, the program will return the value in X\$.

In both cases, a check is made for 'GET'ting the correct type of variable.

If you call 'GET' number and a string value is the next element an error will result and the program aborts.

The order in which the user should 'GET' variable should be the same order as how the programmer had 'PUT' the variables.

## Read A Data Record

After the user has opened the input file, data records may be read. This routine makes data available to the program to 'GET' strings or variables.

An error occurs if the user attempts to read past end-of-file, or read from a closed file.

---

## ERROR MESSAGES

Attempting to close a file that was not opened

A file must be opened before a close command can be executed - check program Logic.

Attempting to 'GET' a string/number before a read

The user is attempting to extract data from the input record before any data way made available to the program.

Attempting to read from an unopened file

The user must open the input file before a read data record can be performed.

Attempting to read past end of file

The user has failed to check properly for the end of file condition. End of file exists when PEEK(28680)=2. Another read should not be performed after the byte is set to 2.

Attempted write to an unopened file

The user must first execute an 'Open Output File' routine to initialize the header REC.

Header label not found

The first detected record was not a header record on the input tape. Check for proper positioning of the tape,

Note: Only tapes created from the program output routine should be used as input to another or the same program.



### Input file already opened

The user is attempting to open the input file without an intervening close statement.

### Label error - Label found = NN

The label the user has specified does not match the label on the tape. 'NN' specifies what the open routine found on the tape. The user will be asked if he wishes to continue. Only a 'Y' response will be accepted in order to continue. All other responses will result in the program aborting.

### Maximum record length exceeded

The sum of all the characters has exceeded 255 characters.

Note: The 255 character limit applies to all control characters embedded within the output record.

### Output file already opened

The user has attempted to open the output file without an intervening 'CLSOE OUTPUT FILE' command.

### Type conflict exists

Two conditions can cause this message.

1. The user has requested a 'GET' string and a numeric field is next or he has requested a 'GET' number and a string field was next.
2. The user has requested a 'GET' string of 'GET' number and there are no remaining fields left in the input string area. The user should check the format specified against the tape format.

## PROGRAM LISTING WITH COMMENTS

### PROGRAM LISTING

1	<u>GOSUB 65000</u>	<i>Required Before any Tape I/O Instructions</i>
10	PRINT "1. Open file as output	7. Put String"
20	PRINT "2. Open file as input	8. Write data record"
30	PRINT "3. Open file as extend	9. Get next number"



```
40      PRINT "4.  Close input file      10.  Get next string"
50      PRINT "5.  Close output file     11.  Read data record"
60      PRINT "6.  Put number"
65      PRINT
70      INPUT "Enter the number of the function or -1 to stop ";C
75      IF C=-1 THEN STOP
80      ONC GOTO 2000,2020,2030,2040,2050,2060,2070,2080,2090,
           2100,2110
90      PRINT "INVALID CHOICE":GOTO 10
```

*Lines 70-90 Get a User Choice and Jump to Terminal Prompting Sections.*

---

```
2000      REM*****OPEN FILE AS OUTPUT
2002      PRINT"READY OUTPUT DRIVE AND HIT (RET) TO CONTINUE":PAUSE
2004      LINE INPUT"ENTER OUTPUT LABEL";Y$
2006      GOSUB 64100
2008      PRINT"DONE":GOTO 10

2020      REM*****OPEN FILE AS INPUT
2022      PRINT"READY INPUT DRIVE AND HIT (RET) TO CONTINUE":PAUSE
2024      LINE INPUT"ENTER INPUT LABEL";X$
2026      GOSUB 64130
2028      GOTO 2008

2030      REM*****OPEN FILE AS EXTEND
2032      PRINT"READY INPUT AND OUTPUT DRIVES AND HIT (RET) TO
CONTINUE":PAUSE
2034      LINE INPUT "ENTER INPUT FILE LABEL";Y$
2036      GOSUB 64190
2038      GOTO 2008
```

```
2040 REM*****CLOSE INPUT FILE
2042 PRINT"INPUT FILE IS NOW CLOSED"
2044 GOSUB 64230
2046 GOTO 2008

2050 REM*****CLOSE OUTPUT FILE
2052 PRINT"OUTPUT FILE IS NOW BEING CLOSED"
2054 GOSUB 64240
2056 GOTO 2008

2060 REM*****PUT NUMBER
2062 INPUT"ENTER A NUMBER OR -1 TO STOP ";Y
2064 IF Y=-1GOTO 2008
2066 GOSUB 64260
2068 GOTO 2062

2070 REM*****PUT STRING
2072 LINE INPUT"ENTER A STRING OR 'END' TO STOP";Y$
2074 IF Y$="END"GOTO 2008
2076 GOSUB 64290
2078 GOTO 2072

2080 REM*****WRITE DATA RECORD
2082 PRINT"RECORD BEING WRITTEN"
2084 GOSUB 64310
2086 GOTO 2008

2090 REM*****GET NUMBER
2092 GOSUB 64360
2094 PRINT"NUMBER RETRIEVED=";X
2096 GOTO 2008
```

```
2100 REM*****GET A STRING
2102 GOSUB 64390
2104 PRINT"STRING RETRIEVED=";X$
2106 GOTO 2008

2110 REM*****READ A RED
2112 PRINT"READING RECORD"
2114 GOSUB 64440
2116 GOTO 2008

64000 REM READ A RECORD
64005 POKE 17990,177:POKE 17991,1 Set Up USR for PAM Load
64010 Z$=" ":Z9=USR(0) Read From Tape
64015 FOR Z8=28684 TO 28683+PEEK(28683)
64020 Z$=Z$+CHR$(PEEK(Z8)) Collect Character From the Buffer
64025 NEXT Z8
64030 RETURN

64040 REM*****WRITE A RECORD
64045 Z9=LEN(Z$):POKE 28683,Z9 Store String Length
64050 FOR Z8=1 TO Z9
64055 POKE 28683+Z8,ASC(MID$(Z$,Z8,1)) Store Char String into Buffer
64060 NEXT Z8
64065 POKE 17990,252:POKE 17991,1 Set up USR for PAM Dump
64070 POKE 8192,10:POKE 8193,112 Beginning Dump ADR
64075 POKE 8212,12=POKE 8213,113 Ending Dump ADR
64080 Z9=USR(0) Write to Tape
64090 RETURN
```

```
64100 REM*****OPEN FILE OUTPUT
64105 IF PEEK(28681)+0 GOTO 64115 If File is Closed. Continue
64110 PRINT Z1$:STOP Error - File Open Already
64115 POKE 28681,1:POKE 28682,0 Set File Status to Onen
Set Rec Type to Header
64120 Z$=Y$:GOSUB 64040:Y9$=" " Write out Header Rec
64125 RETURN

64130 REM*****OPEN FILE INPUT
64135 IF PEEK(28680)=0 GOTO 64145 Check for Already Open File
64140 PRINT Z2$:STOP File Open Error
64145 POKE 28680,1 Set File to Open Status
64150 GOSUB 64000 Get a Record
64155 IF PEEK(28682)=0 GOTO 64165 Check for Header Rec
64160 PRINT Z3$:STOP Not Header
64165 X9=-1:IF Z$=X$ THEN RETURN Label Error?
64170 PRINT Z4$:Z$
64175 LINE INPUT"CONTINUE?Y/N";X0$ See if User Accepts Error
64180 IF X0$="Y"THEN RETURN
64185 STOP

64190 REM*****OPEN FILE EXTENDED
64195 X$=Y$:GOSUB 64130 Open Input File
64200 GOSUB 64100 Open Output File
64205 GOSUB 64440 Read A Rec
64210 IF PEEK(28680)=2 GOTO 64225 EOF?
64212 Y9$=X9$ Move to Output Area
64215 GOSUB 64310 Write Data Rec
64220 GOTO 64205 Loop Until EOF
64225 GOSUB 64230:Y$=" " :RETURN Close Input File
```

```
64230 REM*****CLOSE INPUT FILE
64231 IF PEEK(28680)=1 OR PEEK(28680)=2 GOTO 64235
64232 PRINT X2$:STOP Error If Already Opened
64235 POKE 28680,0:RETURN Set Status to Close

64240 REM*****CLOSE OUTPUT FILE
64241 IF PEEK(28681)=1 GOTO 64245 Check to See if Open
64242 PRINT X2$:STOP
64245 Z$=Z0$:GOSUB 64040 EOF-Write to Tape
64250 POKE 28681,0:RETURN Set Status to Close

64260 REM PUT A NUMBER
64265 Y8$=STR$(Y) Make No A String
64270 IF LEN(Y8$)+LEN(Y9$)+3 <=255 GOTO 64280 Check for Overflow
64275 PRINT Z5$:STOP
64280 Y9$=Y9$+CHR$(28)+Y8$ Append to Output String
64285 RETURN

64290 REM*****PUT A STRING
64295 IF LEN(Y9$)+LEN(Y$)+3 <=255 GOTO 64305 Check for Overflow
64300 PRINT Z5$:STOP
64305 Y9$=Y9$+CHR$(29)+Y$:RETURN Append to Output String

64310 REM*****WRITE A DATA RECORD
64315 IF LEN(Y9$)+3 <=255 GOTO 64325 Check for Overflow
64320 PRINT Z5$:STOP
64325 Y9$=Y9$+CHR$(3) Append ETX Character
64330 IF PEEK(28681)=1 GOTO 64340 Check for File Open
64335 PRINT Z6$:STOP
64340 POKE 28682,2 Set Record Type to Data
64345 Z$=Y9$:GOSUB 64040 Write Record
64350 Y9$=" ":RETURN
```



```
64360 REM*****GET A NUMBER
64361 IF X9 >=1 GOTO 64365      Check for Available Data
64362 PRINT X3$:STOP          Read First ERR
64365 IF MID$(X9$,X9,1)=CHR$(28)GOTO 64375  Conflict?
64370 PRINT Z7$:STOP
64375 X$=" ":GOSUB 64410       Append Char and Build String
64380 X=VAL(X$):RETURN        Convert to Numeric

64390 REM*****GET A STRING
64391 IF X9 >=1 GOTO 64395      Check for Available Data
64392 PRINT X3$:STOP
64395 IF MID$(X9$,X9,1)=CHAR(29)GOTO 64405  Conflict?
64400 PRINT Z7$:STOP
64405 X$=" ":GOSUB 64410:RETURN  Build String

64410 REM*****COLLECT A STRING
64415 X9=X9+1                  Increment Input Pointer
64420 X8$=MID$(X9$,X9,1)       Get Next Char
64425 IF X8$=CHR$(28)OR X8$=CHR$(29)OR X8$=CHR$(3)
      THEN:RETURN              Is It a Delimiter?
64430 X$=X$+X8$:GOTO 64415     Build String: Get Another Char

64440 REM*****READ A DATA RECORD
64445 IF PEEK(28680)=1 GOTO 64451      Check File Status
64450 PRINT X1$:STOP
64451 IF PEEK(28680) <>2 GOTO 64455     Check for EOF Read
64452 PRINT X4$:STOP
64455 GOSUB 64000                 Read A Record
64460 X9$=Z$
64465 IF X9$=Z0$:THEN POKE(28680)2  If EOF Set Flag In File Status
64470 X9=1:RETURN
```

```
65000    REM*****MESSAGES AND CONSTANTS
65005    Z0$="1EOF-H8"
65010    Z1$="**OUTPUT FILE ALREADY OPENED**"
65015    Z2$="**INPUT FILE ALREADY OPENED**"
65020    Z3$="**HEADER LABEL NOT FOUND**"
65025    Z4$="**LABEL ERROR-LABEL FOUND="
65030    Z5$="**MAXIMUM RECORD LENGTH EXCEEDED**"
65035    Z6$="**ATTEMPTED WRITE TO UNOPENED FILE**"
65040    Z7$="**TYPE CONFLICTS EXIST**"
65045    X1$="**ATTEMPTING TO READ FROM AN UNOPENED FILE**"
65046    X2$="**ATTEMPTING TO CLOSE A FILE THAT WAS NOT OPENED**"
65047    X3$="**ATTEMPTING TO GET A STRING/NUMBER BEFORE A READ**"
65048    X4$="**ATTEMPTING TO READ PAST END OF FILE**"
65050    POKE 28260,0:POKE 28261,0    Set File Status to Close
65055    RETURN
```

## FILEMAINT/MULTISORT

Roseman

The file maintenance system enables the user to create, update and sort a file placed on tape. The system consists of two programs, FILEMAINT and MULTISORT. The sort is run as a separate program in order to make as much memory available to it as possible.

The tape I/O routines are those of Mr. Bellinger\* (to whom I will be forever grateful). While these routines are rather slow, they do seem to work quite well. They offer program controlled sequential I/O operations on files too large to be read into memory.

When you run the FILEMAINT program you are asked which of four options you desire. The options are:

1. Establish new file.
2. Insert additional items to an existing file.
3. Make changes or deletions to an existing file.
4. List out an existing file. A message is generated requesting entry of the option desired.

Each option operates as follows:

### 1. ESTABLISH NEW FILE

A message is generated requesting the number of fields desired in the record and the name of each field. Following these entries you will be directed to ready an output tape (please use a clean one), and press space bar when ready.

The program will now accept entries to the record. You will be prompted in these entries by the names previously supplied to the program.

After entering the last data field in the record you are asked to check it for mistakes. Press return if record is OK. If you discover that you entered some wrong information you simply key in the number of the data field in error and reenter the data. Again you will be asked is the record is OK; if it is, hit carriage return and the record is written out to tape in the following format:

/DATA1/DATA2/DATA3/ - - - - - /DATA<sub>n</sub>

The slashes are used as delineators. Note that the total length of a record cannot exceed 256 characters including slashes.

The system now asks if you are finished. If not, you are again prompted to enter a new record. If you are finished, the system writes a trailer record to the tape file. The trailer record consists of an asterisk (\*). After the trailer is written, the program asks if you wish to continue using the FILEMAINT system. At this point I usually like to take a look at the file and check it for errors. (I suspect my real motive is just to admire it.) If you wish to look at the file, select option number four (4).

## 2. INSERT ADDITIONAL ITEMS TO AN EXISTING FILE

When this option is selected you are instructed to ready an input and output tape and to press the space bar when ready. The system then proceeds to read each record off the input tape and write it to the output tape. When the end of file trailer is detected on the input tape, the program reverts to the same routine it used in option number one (1) where you first established the file.

You are asked to name the input fields so the system can prompt you through your entries. However, you are not required to input the number of data fields because the system has already established this while it read the input file, i.e., all records in the file must have the same number of data fields; a count of slashes in any given record determines the number of fields for all records. As in option number one (1), the records are written to the output tape as you enter and verify them. After each record entry, you are asked to determine whether or not you are finished or wish to continue.

## 3. MAKE CHANGES OR DELETIONS TO AN EXISTING FILE

This option invokes a very flexible routine which allows you to correct or delete entire records or only certain fields within the records or any combination of the above.



When you choose this option, you are asked to enter the field number to be used as the search argument. Next you must enter the search argument itself. You are then asked to mount tapes and to press space bar when ready. The input tape is read and the content of the field you selected is compared against the search argument you entered. If they are not equal then the record is written to output tape and a new record is read from the input tape. The above procedure is repeated until the proper record is found. When the record to be corrected is found, it is printed out and you are asked to enter the field number to be corrected or to enter zero (0) to delete or correct the whole record. Assuming you enter (0), you are then asked to hit carriage return to delete the record or to reenter the entire record. If you select a field number, you are then asked to reenter the field. In either case, the system asks if you are finished with corrections to this record. If so, the changed record is written to the output tape; if not, you are again asked to enter a field number and the procedure is repeated. Once the record has been written to the output tape, a prompt will ask whether or not you desire to make any more changes.

If your file is already sorted (e.g., on last name) then you can save time by entering your corrections in the same order. If desired, the program will continue to search the input tape for additional records to be corrected or deleted. When no more changes are to be made, the remaining input records are written to the output tape.

#### 4. LIST OUT AN EXISTING FILE

When this option is chosen, you are instructed to ready the input tape and press the space bar when ready. The program reads the input tape and lists it out record by record, i.e., read record, print record, read record, print record, etc. When the file trailer (\*) is encountered, you are again asked if you desire to end or continue. Prompts direct proper responses for the end or continue function. Tape copies can also be made using this option.



## GENERAL PROGRAM COMMENTS:

1. In order to use Bellinger's tape I/O routines, it is necessary that BASIC be reconfigured so that the "HIGH MEMORY" limit is lowered 256 bytes. The author's system presently has 20K (HIGH MEMORY = 28672), just to be on the safe side I lowered it 300 bytes to 28372. Statement #100 in the program listing should be changed to the user's reconfigured high memory limit.
2. FILEMAINT was written in EXTENDED BASIC Rec. 10.02.01. In this revision, USRFCN is located in memory positions 17990 and 17991. If the user has a different revision, then statement #65010, 65020, 65180 and 65190 must be changed to that revisions USRFCN address (in decimal). The USRFCN address can be found in the table at the rear of the BASIC manual.

The second part of the FILEMAINT system is a program which enables the user to sort the tape file created by the FILEMAINT program. In using the MULTISORT program, up to ten (10) sort fields can be specified. These fields are specified from major to minor, for example, in the record below suppose we wish to sort on salary by group in a given region. We enter the sort fields as 5NA, 7NA, 3ND (where A is for ascending and D is for descending).

#1	#2	#3	#4	#5	#6	#7
/LAST NAME/FIRST NAME/SALARY/ADDRESS/REGION/SSN/GROUP						

The sort operates as follows:

1. The user is asked to input the number of records in the input file. This may be overestimated but should never be underestimated. Try not to overestimate too much as this costs memory.
2. The next input is the number of fields to be sorted on. In the example above, this would be three (3).
3. Now the field numbers are input one at a time. Again in the example, these would be 5NA, 7NA, 3ND.

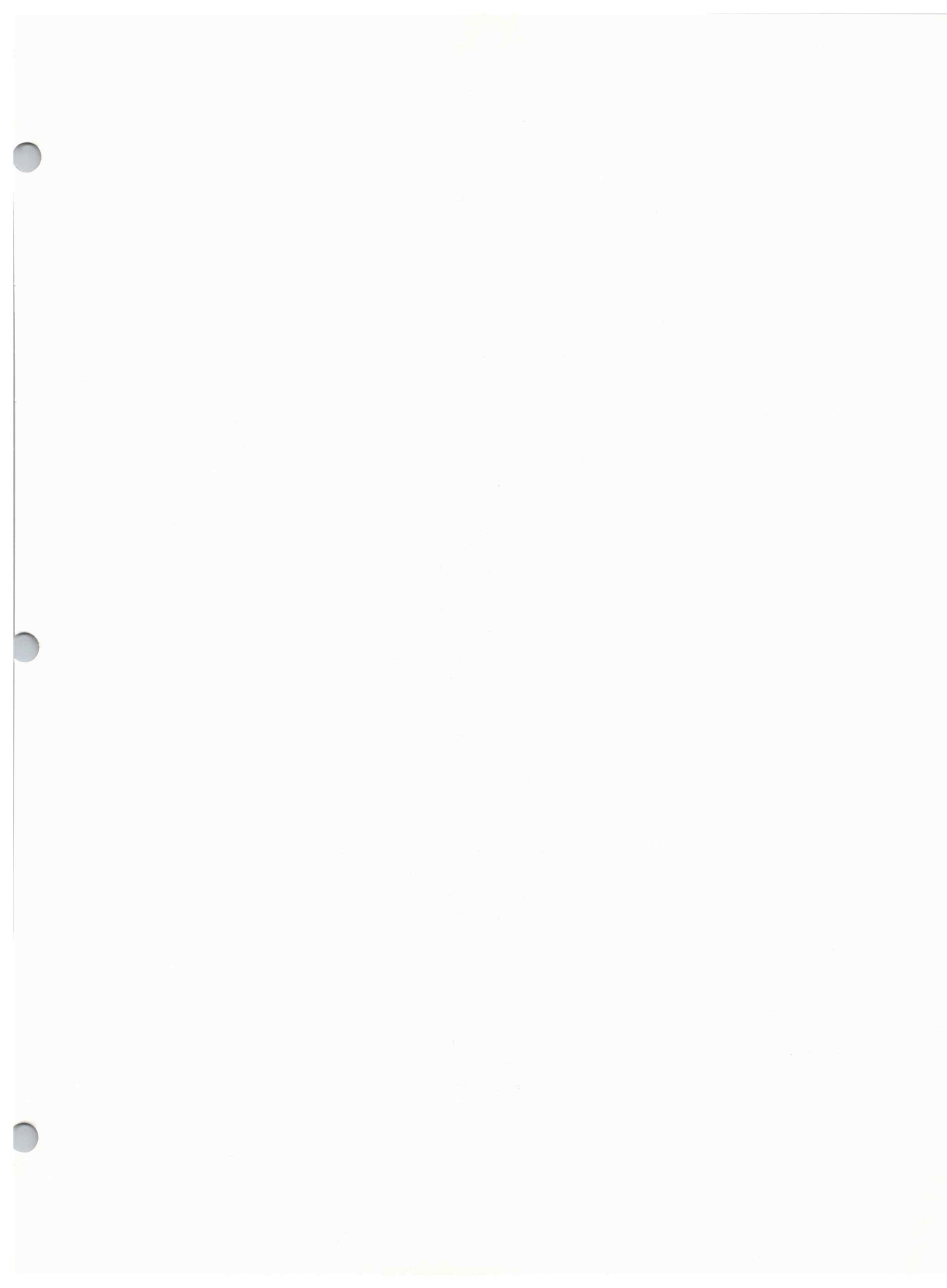
4. The program now asks you to ready the input tape and to press the space bar when ready.
5. As the program reads the input tape, it builds a table in memory containing only the data contained in the sort fields of each record and the record number. NOTE: If this table exceeds your available memory, then the sort cannot be run. In this case you must reduce the number of sort fields specified in (2) above.
6. The above table is now sorted.
7. The program next asks you to ready an output tape and to press the space bar when ready.
8. The program predetermines the number of whole records it can read into memory. It does this by dividing your available memory by an average record length which was determined during the first pass of the input tape. The number of records on the input tape was also determined during the first pass. The number of records on the input tape is divided by the estimated number of whole records that can be read into memory to determine the minimum number of passes of the input tape that will be required to complete the sort. The minimum number of passes is then printed out.
9. You are now asked to rewind the input tape and ready it to be read again.
10. The program now tries to match records off the input tape against the sorted table. When a match is found, the record is placed in memory. This process continues until we run short of memory or we encounter an end of file (\*).
11. The table of matching records is now written to the output tape and the user is again asked to rewind the input tape for the next pass. This process continues until we satisfy all records in the sorted table. The same general program comments apply to this program as to FILEMAINT.

Hardware requirements for this system are:

H8 with 16K

Terminal

Two (2) tape recorders



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: FILE MAINTAINANCE - ROSEMAN

```
00050 REM *** SET GLOBAL CONSTANTS *** -----
00100 M=28372
00200 DEF FN L(X)=X-256*INT(X/256)
00300 DEF FN U(X)=INT(X/256)
00400 D=0
00600 C7=0
00640 REM *** PRINT OPTIONS *** -----
00660 PRINT "FILE MAINTENANCE OFFERS YOU THE FOLLOWING OPTIONS:"
00700 PRINT "-----"
00800 PRINT
00900 PRINT "1.- ESTABLISH NEW FILE"
01000 PRINT "2.- INSERT ADDITIONAL ITEMS TO AN EXISTING FILE"
01100 PRINT "3.- MAKE CHANGES OR DELETIONS TO AN EXISTING FILE"
01200 PRINT "4.- LIST OUT AN EXISTING FILE"
01300 PRINT :PRINT
01350 REM *** ENTER DESIRED OPTION *** -----
01400 INPUT "ENTER OPTION YOU DESIRE (1,2,3,OR 4) ";O
01500 IF O>4 OR O<1 GOTO 1400
01600 ON O GOTO 1700,5700,7700,13900
01650 REM *** NEW FILE RTN *** -----
-----
01700 U=1
01800 INPUT "HOW MANY FIELDS ARE THERE IN RECORD ";F
01850 F1=F
01900 IF D=1 THEN GOSUB 15000
02000 D=1
02050 F=F1
02100 DIM T$(F),D$(F)
02150 REM *** ESTABLISH FIELD NAMES FOR PROMPTS *** -----
02200 FOR I=1 TO F
02300 PRINT "ENTER TITLE OF FIELD #";I
02400 LINE INPUT ;T$(I):T$(I)=T$(I)+" ": "
02500 NEXT I
02600 IF U=2 GOTO 2900
02700 PRINT "READY OUTPUT TAPE --PRESS SPACE BAR WHEN READY--":PA
USE
02800 A=0
02850 REM *** CLEAR SCREEN *** -----
02900 PRINT :PRINT :PRINT :PRINT :PRINT :PRINT
03000 PRINT :PRINT :PRINT :PRINT :PRINT :PRINT
03100 PRINT "-- RECORD # ";A+1;"--"
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: FILE MAINTAINANCE - ROSEMAN < CONT'D >

```
03150 REM *** ENTER RECORD *** -----
03200 FOR I=1 TO F
03300 PRINT T$(I)
03350 LINE INPUT ;D$(I)
03400 NEXT I
03500 PRINT "PRESS RETURN IF RECORD IS OK"
03600 LINE INPUT "ENTER FIELD # TO BE CORRECTED IF NOT ";E$
03700 IF E$="" GOTO 4100
03800 IF VAL(E$)<1 OR VAL(E$)>F GOTO 3500
03900 PRINT T$(VAL(E$))
03950 LINE INPUT ;D$(VAL(E$))
04000 GOTO 3500
04050 REM *** CONSTRUCT OUTPUT RECORD *** -----
04100 Z$=""
04200 FOR I= 1 TO F
04300 Z#=Z#+"/"+D$(I)
04400 NEXT I
04600 GOSUB 65110
04700 LINE INPUT "PRESS RETURN TO CONTINUE ENTRY, ENTER (F) IF FI
NISHED ";F$
04800 IF F$="" GOTO 3100
04900 IF F$<>"F" GOTO 4700
04950 REM *** WRITE TRAILER RECORD TO FILE *** -----
05000 Z$="*"
05100 GOSUB 65110
05150 IF S1=0 AND V=3 THEN PRINT "SEARCH ARGUMENT OF ";S$;" NOT F
OUND IN FILE"
05200 LINE INPUT "ENTER (E) TO END PROGRAM,ENTER (C) TO CONTINUE
";U$
05300 IF U$="C" THEN PRINT "-----";A;"RECORDS WRITTEN -----":GO
TO 600
05400 IF U$<>"E" GOTO 5200
05500 PRINT "-----";A;"RECORDS WRITTEN -----"
05600 END
05650 REM *** INSERT ADDITIONAL ITEMS RTN *** -----
-----
05675 IF C7=1 GOTO 5900
05700 PRINT "READY INPUT AND OUTPUT TAPES --PRESS SPACE BAR WHEN
READY--":PAUSE
05800 A=0
05900 C7=0
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: FILE MAINTAINANCE - ROSEMAN < CONT'D >

```
06000 GOSUB 65000
06050 REM *** SEARCH FOR END OF FILE *** -----
06100 IF Z$="*" GOTO 6900
06200 IF V<>3GOTO 6600
06300 GOSUB 8100
06400 IF C7=1 GOTO 7700
06600 Z1$=Z$
06700 GOSUB 65110
06800 GOTO 6000
06850 REM *** COUNT FIELDS IN RECORD *** -----
06900 IF V=3 GOTO 5100
07000 Y=LEN(Z1$)
07100 F1=0
07200 V=2
07300 FOR I=1 TO Y
07400 IF MID$(Z1$,I,1)="/" THEN F1=F1+1
07500 NEXT I
07600 GOTO 1900
07650 REM *** CHANGE/DELETE RTN *** -----
-----
07700 INPUT "ENTER FIELD # YOU WISH TO KEY ON IN RECORD TO BE CHA
NGED ";K
07800 LINE INPUT "ENTER SEARCH KEY FOR ABOVE FIELD ";S$
07900 V=3
08000 GOTO 5650
08100 REM *** TEST FIELD RTN *** -----
08200 IF C7=2 THEN RETURN
08250 S1=0
08300 C1=0
08400 C2=1
08500 Y1=LEN(Z$)
08600 Z2$=""
08700 IF MID$(Z$,C2,1)="/" GOTO 9100
08800 C2=C2+1
08900 IF C2>Y1 THEN RETURN
09000 GOTO 8700
09100 C1=C1+1
09150 REM *** CORRECT SEARCH FIELD ? *** -----
09200 IF C1<>K GOTO 8800
09300 C2=C2+1
09400 IF C2>Y1 GOTO 9800
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: FILE MAINTAINANCE - ROSEMAN < CONT'D >

```
09500 IF MID$(Z$,C2,1)="/" GOTO 9800
09600 Z2$=Z2$+MID$(Z$,C2,1)
09700 GOTO 9300
09750 REM *** DOES FIELD MATCH SEARCH KEY ? *** -----
09800 IF S$=Z2$ THEN S1=1:GOTO 10000
09900 RETURN
10000 PRINT Z$
10100 PRINT "ENTER (0) TO CHANGE OR DELETE WHOLE RECORD"
10200 INPUT "ENTER FIELD # TO CHANGE OR DELETE THAT FIELD ";G
10250 IF G<0 GOTO 10100
10300 IF G<>0 GOTO 10700
10350 REM *** CHANGE OR DELETE WHOLE RECORD *** -----
10400 PRINT "INPUT NEW RECORD OR HIT RETURN KEY TO DELETE"
10500 LINE INPUT "--";Z$
10600 GOTO 12100
10650 REM *** COPY OLD RECORD UP TO CHANGE FIELD *** ---
10700 C1=0
10800 C2=1
10900 Z3$=""
11000 Z3$=Z3$+MID$(Z$,C2,1)
11100 IF MID$(Z$,C2,1)="/" GOTO 12500
11200 C2=C2+1
11300 IF C2>V1 GOTO 11600
11400 Z3$=Z3$+MID$(Z$,C2,1)
11500 GOTO 11100
11600 PRINT "FIELD";G;"NOT FOUND"
11650 REM *** ANY MORE CHANGES TO THIS RECORD ? *** ----
11700 LINE INPUT "ENTER (N) FOR MORE CHANGES TO THIS RECORD,(N) F
OR NONE ";U$
11800 IF U$="N" GOTO 12100
11900 GOTO 11700
12050 REM *** ANY MORE CHANGES TO THIS FILE ? *** -----
12100 LINE INPUT "ENTER (C) FOR MORE CHANGES TO THIS FILE,(N) FOR
NONE ";U$
12200 IF U$="C" THEN C7=1:GOSUB 65110:RETURN
12300 IF U$="N" THEN C7=2:RETURN
12400 GOTO 12100
12500 C1=C1+1
12550 REM *** TEST FOR CHANGE/DELETE FIELD *** -----
12600 IF C1<>G GOTO 11200
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: FILE MAINTAINANCE - ROSEMAN < CONT'D >

```
12650 REM *** ENTER AND COPY CHANGE *** -----
12700 LINE INPUT "ENTER CHANGE ";Q$
12800 Z3$=Z3$+Q$
12850 REM *** COPY REST OF RECORD *** -----
12900 C2=C2+1
13000 IF C2>Y1 GOTO 13700
13100 IF MID$(Z$,C2,1)="/" GOTO 13300
13200 GOTO 12900
13300 Z3$=Z3$+MID$(Z$,C2,1)
13400 C2=C2+1
13500 IF C2>Y1 GOTO 13700
13600 GOTO 13300
13700 Z$=Z3$
13750 Y1=LEN(Z$)
13800 GOTO 11700
13900 REM *** LIST RTN *** -----
-----
14000 PRINT "MOUNT INPUT TAPE --PRESS SPACE BAR WHEN READY--":PAU
SE
14010 PRINT "IF YOU WISH TO COPY THIS FILE WHILE IT IS LISTED,THE
N ENTER (C).",
14020 LINE INPUT "IF NOT THEN HIT RETURN ";Q$
14030 IF Q$="C" THEN PRINT "MOUNT OUTPUT TAPE -- PRESS SPACE BAR
WHEN READY --":PAUSE
14100 GOSUB 65000
14110 IF Q$="C" THEN GOSUB 65110
14200 IF Z$="*" GOTO 14500
14300 PRINT Z$
14400 GOTO 14100
14500 PRINT :PRINT
14600 LINE INPUT "ENTER (C) TO CONTINUE,(E) TO END ";P$
14700 IF P$="C" GOTO 600
14800 IF P$<>"E" GOTO 14600
14900 END
15000 REM *** CLEARS TABLES *** -----
15100 FOR I=1 TO F1
15200 T$(I)=""
15300 D$(I)=""
15400 NEXT I
15500 CLEAR T$(
15600 CLEAR D$(
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: FILE MAINTAINANCE - ROSEMAN < CONT'D >

```

15800 RETURN
65000 REM *** TAPE READ RTN ***
65010 POKE 17990,177
65020 POKE 17991,1
65030 POKE 8192,FN L(M)
65040 POKE 8193,FN U(M)
65050 Z=USR(0)
65060 Z$=""
65070 FOR Z=M+1 TO M+PEEK(M)
65080 Z$=Z$+CHR$(PEEK(Z))
65090 NEXT Z
65100 RETURN
65110 REM *** TAPE WRITE RTN ***
65120 Z0=LEN(Z$)
65125 IF Z0=0 THEN RETURN
65127 A=A+1
65130 POKE M,Z0
65140 FOR Z=1 TO Z0
65150 POKE M+Z,ASC(MID$(Z$,Z,1))
65160 NEXT Z
65170 Z=M+Z0
65180 POKE 17990,252
65190 POKE 17991,1
65200 POKE 8192,FN L(M)
65210 POKE 8193,FN U(M)
65220 POKE 8212,FN L(Z)
65230 POKE 8213,FN U(Z)
65240 Z=USR(0)
65250 RETURN

```

10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTI SORT - ROSEMAN

```
00095 REM *** INPUT AND INITIALIZATION SECTION ***
00096 DIM S(10),B(10,2),X$(10),V$(10)
00100 M=28372
00200 I1=1
00300 A=0
00400 B=0
00500 M1=346
00600 K=0
00700 P=0
00800 C=1
00900 A1=0
01000 B1=1
01100 L6=1
01200 L8=0
01300 DEF FN L(X)=X-256*INT(X/256)
01400 DEF FN U(X)=INT(X/256)
01500 INPUT "INPUT # OF RECORDS ON INPUT FILE (CAN BE OVERESTIMAT
ED) ";R
01600 INPUT "INPUT # OF SORT FIELDS (10 OR LESS) ";N
01700 FOR I=1 TO N
01800 PRINT "ENTER FIELD #, TYPE (X-ALPHANUMERIC OR N-NUMERIC),"
01850 PRINT "AND SORT SEQUENCE (A-ASCENDING OR D-DESCENDING), FOR
SORT FIELD";I
01900 LINE INPUT ":";F2$
01910 IF VAL(MID$(F2$,1,1))<1 OR VAL(MID$(F2$,1,1))>10 GOTO 1800
01920 IF MID$(F2$,2,1)<>"X" AND MID$(F2$,2,1)<>"N" GOTO 1800
01930 IF MID$(F2$,3,1)<>"A" AND MID$(F2$,3,1)<>"D" GOTO 1800
01940 S(I)=VAL(MID$(F2$,1,1))
01950 X$(I)=MID$(F2$,2,1)
01960 V$(I)=MID$(F2$,3,1)
02000 NEXT I
02050 REM *** CONSTRUCT TABLE OF KEY FIELDS ***
02100 DIM N1(R),T$(R)
02200 PRINT "READY INPUT TAPE --PRESS SPACE BAR WHEN READY--":PAU
SE
02300 GOSUB 65000
02400 L8=L8+LEN(T$(A))
02500 IF M1-(50+4*R+L8)<0 THEN PRINT "SORT TABLE > MEM AT REC ";A
:END
02600 IF MID$(Z$,1,1)="*" GOTO 5075
02700 A=A+1
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTI SORT - ROSEMAN < CONT'D >

```
02800 N1(A)=A
02900 Y=LEN(Z$)
03000 A1=A1+Y+2
03100 FOR I=1 TO N
03300 L7=0
03400 FOR J=1 TO Y
03500 IF MID$(Z$,J,1)<>"/" GOTO 4900
03600 L7=L7+1
03700 IF L7<>5(I) GOTO 4900
03750 T$(A)=T$(A)+"/"
03800 J=J+1
03900 IF J>Y GOTO 4600
04000 IF MID$(Z$,J,1)="/" GOTO 4600
04100 T$(A)=T$(A)+MID$(Z$,J,1)
04200 J=J+1
04300 IF J>Y GOTO 5000
04400 IF MID$(Z$,J,1)="/" GOTO 5000
04500 GOTO 4100
04600 T$(A)=T$(A)+" "
04700 PRINT "NULL DATA IN RECORD #";A;"FOR SORT FIELD";I
04800 GOTO 5000
04900 NEXT J
05000 NEXT I
05050 GOTO 2300
05070 REM *** SORT TABLE ***
05075 REM (NOP)
05100 X1=0:GOSUB 15000
06350 REM *** WRITE OUTPUT TAPE ***
06400 PRINT "MOUNT OUTPUT TAPE --PRESS SPACE BAR WHEN READY--":PA
USE
06500 P=P+1
06550 IF M1-8*R-50<0 GOTO 6570
06560 GOTO 6700
06570 PRINT "TOO MANY RECORDS FOR SORT TO HANDLE"
06580 PRINT "YOU NEED";50+8*R-M1;"MORE BYTES"
06700 P1=INT((A1/(M1-8*A-50)+1))
06800 Q=INT(A/P1+1)
06900 PRINT "THERE WILL BE A MINIMUM OF";P1;"PASSES"
07000 FOR D6=1 TO R
07010 T$(D6)=" "
07020 NEXT D6
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTI SORT - ROSEMAN < CONT'D >

```
07030 CLEAR T$( : DIM T$(Q+1)
07100 PRINT "REWIND INPUT TAPE AND READY IT FOR PASS";P
07200 PRINT "--PRESS SPACE BAR WHEN READY--":PAUSE
07300 B2=0
07400 L2=0
07500 L3=0
07600 GOSUB 65000
07700 IF MID$(Z$,1,1)="*" GOTO 9100
07800 B2=B2+1
07900 IF B1+Q>A THEN Q=A-B1+1:L6=2
08000 L4=0
08100 FOR J=B1 TO B1+Q-1
08200 L4=L4+1
08300 IF B2=N1(J) GOTO 8500
08400 NEXT J
08450 GOTO 7600
08500 T$(L4)=Z$
08600 L2=L2+LEN(Z$)+2
08700 L3=L3+1
08800 IF L3=Q GOTO 9100
08900 IF M1-(4*Q+4*R+50+L2)<0 GOTO 9100
09000 GOTO 7600
09100 L5=1
09200 IF LEN(T$(1))=0 GOTO 10060
09300 Z$=T$(L5)
09400 GOSUB 65110
09500 L5=L5+1
09550 B=B+1
09600 IF LEN(T$(L5))=0 GOTO 9800
09650 IF L5>Q GOTO 9800
09700 GOTO 9300
09800 IF L6=2 GOTO 10100
09900 B1=B1+L5-1
09910 FOR D6=1 TO Q+1
09920 T$(D6)=" "
09930 NEXT D6
09940 P=P+1
09950 IF B1>A GOTO 10100
10000 GOTO 7100
10060 Q=Q-2
10070 P=P+1
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTI SORT - ROSEMAN < CONT'D >

```
10080 PRINT "Q REDUCED TO";Q
10090 GOTO 7000
10100 Z$="*"
10200 GOSUB 65110
10300 PRINT "END OF SORT, REWIND ALL TAPES"
10400 PRINT A;"RECORDS INPUT TO SORT";B;"RECORDS OUTPUT"
10500 END
10600 REM ***SHIFT RTN ***
10700 T1$=T$(J)
10800 T$(J)=T$(I)
10900 T$(I)=T1$
11000 U6=N1(J)
11100 N1(J)=N1(I)
11200 N1(I)=U6
11250 F1$=F2$
11255 F2$="0"
11300 RETURN
11400 REM *** FIND FIELD RTN ***
11500 F3$=""
11550 IF LEN(T$(Q))=0 THEN RETURN
11600 T=0
11700 T1=1
12000 IF MID$(T$(Q),T1,1)="/" GOTO 12300
12100 T1=T1+1
12200 GOTO 12000
12300 T=T+1
12400 IF T=L GOTO 12600
12500 GOTO 12100
12600 T1=T1+1
12700 F3$=F3$+MID$(T$(Q),T1,1)
12800 T1=T1+1
12900 IF T1>LEN(T$(Q)) GOTO 13200
13000 IF MID$(T$(Q),T1,1)="/" GOTO 13200
13100 GOTO 12700
13200 IF S=1 THEN F1$=F3$:RETURN
13300 F2$=F3$:RETURN
15000 FOR I=1 TO N
15100 B(I,1)=1
15200 NEXT I
15300 B(0,2)=A
15400 L=1
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTI SORT - ROSEMAN < CONT'D >

```
15500 N1=1
15600 N2=A
15700 IF N1=N2 GOTO 15900
15800 GOSUB 16500
15900 IF L=N THEN GOSUB 19400
15950 IF X1=1 THEN RETURN
16000 GOSUB 18100
16100 N1=B(L,1)
16200 N2=B(L,2)
16300 L=K1
16400 GOTO 15700
16500 REM *** SUBROUTINE SORT ***
16600 FOR I=N1 TO N2-1
16700 F1$=""
16800 Q=I
16900 S=1
17000 GOSUB 11500
17100 S=2
17300 FOR J=I+1 TO N2
17400 Q=J
17500 F2$=""
17600 GOSUB 11500
17610 IF X$(L)="X" GOTO 17650
17613 IF U$(L)="A" GOTO 17620
17616 IF VAL(F1$)<VAL(F2$) THEN GOSUB 10700
17619 GOTO 17800
17620 IF VAL(F1$)>VAL(F2$) THEN GOSUB 10700
17630 GOTO 17800
17650 IF U$(L)="A" GOTO 17700
17660 IF F1$<F2$ THEN GOSUB 10700
17670 GOTO 17800
17700 IF F1$>F2$ THEN GOSUB 10700
17800 REM (NOP)
17850 NEXT J
17900 NEXT I
18000 RETURN
18100 REM *** SUBROUTINE COUNT ***
18200 Q=B(L,1)
18300 S=1
18400 GOSUB 11500
18500 H$=F1$
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTI SORT - ROSEMAN < CONT'D >

```
18600 Q=Q+1
18700 IF Q>B(L-1,2) GOTO 19100
18800 GOSUB 11500
18900 IF H$<>F1$ GOTO 19100
19000 GOTO 18600
19100 B(L,2)=Q-1
19200 K1=L+1
19300 RETURN
19400 REM *** SUBROUTINE BACK ***
19500 L=L-1
19600 IF L=0 THEN X1=1:RETURN
19700 B(L,1)=B(L,2)+1
19800 IF B(L,1)>B(L-1,2) GOTO 19500
19900 RETURN
65000 REM *** TAPE READ RTN ***
65010 POKE 17990,177
65020 POKE 17991,1
65030 POKE 8192,FN L(M)
65040 POKE 8193,FN U(M)
65050 Z=USR(0)
65060 Z$=""
65070 FOR Z=M+1 TO M+PEEK(M)
65080 Z$=Z$+CHR$(PEEK(Z))
65090 NEXT Z
65100 RETURN
65110 REM *** TAPE WRITE RTN ***
65115 PRINT Z$
65120 Z0=LEN(Z$)
65125 IF Z0=0 THEN RETURN
65130 POKE M,Z0
65140 FOR Z=1 TO Z0
65150 POKE M+Z,ASC(MID$(Z$,Z,1))
65160 NEXT Z
65170 Z=M+Z0
65180 POKE 17990,252
65190 POKE 17991,1
65200 POKE 8192,FN L(M)
65210 POKE 8193,FN U(M)
65220 POKE 8212,FN L(Z)
65230 POKE 8213,FN U(Z)
65240 Z=USR(0)
65250 RETURN
```

PROGRAM NAME: MINIBUG - HORNE

00010 PRINT "

"

```
00020 PRINT TAB(30);"MINI BUG 8":GOSUB 770
00021 :
00030 PRINT TAB(26);"BY D.F.HORNE 7/78"
00040 PRINT :LINE INPUT " DO YOU WANT INSTRUCTIONS ? Y/N - ";D$
00050 PRINT :IF D$="N" THEN GOTO 360
00060 PRINT " MINI BUG 8 IS DESIGNED TO AID IN EXPERIMENTING WITH
H MACHINE"
00070 PRINT " LANGUAGE . IN THE COMMAND MODE MINI BUG 8 ALLOWS THE
USER TO"
00080 PRINT " READ OR ALTER MEMORY, CLEAR FIELDS OF MEMORY, CONTROL
THE H8"
00090 PRINT " FRONT PANEL DISPLAY, RUN PROGRAMS IN MACHINE LANGUAGE,
AND"
00100 PRINT " EXPLORE MODIFICATIONS TO 'EXTENDED BENTON HARBOR BASIC' ."
00110 PRINT " MINI BUG 8 ACCEPTS EITHER DECIMAL OR OFFSET OCTAL
NOTATION"
00120 PRINT " AND CAN BE USED TO CONVERT FROM ONE TO THE OTHER ."
00130 PRINT :PRINT TAB(7);"- YOU ARE LIMITED ONLY BY YOUR IMAGINATION -"
00140 A1$=" PUSH THE 'SPACE' BAR TO CONTINUE"
00150 PRINT :PRINT A1$:PAUSE :PRINT :PRINT
00160 PRINT " MINI BUG 8 USES THE 'USR()' FUNCTION OF EXTENDED BASIC"
00170 PRINT " AND REQUIRES THAT YOUR PARTICULAR VERSION OF EX. B. H.
BASIC"
00180 PRINT " BE CONFIGURED TO ALLOW FOR MEMORY SPACE ABOVE THE UPPER"
00190 PRINT " LIMIT OF EX. B.H.BASIC ."
00200 PRINT " IT IS SUGGESTED THAT YOU CONFIGURE YOUR DISTRIBUTION
TAPE"
00210 PRINT " WITH AN UPPER MEMORY LIMIT NO LOWER THAN 23700 (DECIMAL)"
```

PROGRAM NAME: MINIBUG - HORNE

```
00220 PRINT " A 16K SYSTEM CONFIGURED AS SUGGESTED WILL PROVIDE T
HE"
00230 PRINT " USER WITH APROX. 750 BYTES RAM ABOVE THE SPACE OCCU
PIED BY"
00240 PRINT " EX. B.H.BASIC AND MINI BUG 8 FOR MACHINE LANGUAGE P
ROGRAMS"
00250 PRINT :PRINT A1$:PAUSE :PRINT :PRINT
00260 PRINT TAB(6);"ENTER THE FOLLOWING CODES FOR THE INDICATED
OPERATION"
00270 PRINT
00280 PRINT " ZERO - CLEARS A FIELD OF MEMORY"
00290 PRINT " OD - CONVERTS OFFSET OCTAL TO DECIMAL NOTATION"
00300 PRINT " DO - CONVERTS DECIMAL TO OFFSET OCTAL NOTATION"
00310 PRINT " LM - PROVIDES A LIST OF DATA IN MEMORY"
00320 PRINT " AM - ALLOWS USER TO ALTER DATA IN MEMORY"
00330 PRINT " GO - ALLOWS USER TO RUN A MACHINE LANGUAGE PROGRA
M"
00340 PRINT " H8 - PERMITS CONTROL OF H8 FRONT PANEL DISPLAY"
00350 PRINT
00360 PRINT :LINE INPUT " DESIRED OPERATION ? - ";O$
00370 IF O$="ZERO"THEN GOTO 1780
00380 IF O$="OD"THEN GOTO 900
00390 IF O$="DO"THEN GOTO 820
00400 IF O$="LM"THEN GOTO 1300
00410 IF O$="AM"THEN GOTO 1550
00420 IF O$="GO"THEN GOTO 1940
00430 IF O$="H8" THEN GOTO 980
00440 LINE INPUT " DO YOU NEED THE LIST OF COMMANDS ? Y/N -
";O1$
00450 IF O1$="Y" THEN GOTO 260
00460 GOTO 360
00470 IF B$="O"THEN GOTO 500
00480 IF VAL(T$)>65535 THEN PRINT " ERROR - NONEXISTENT NUMBER"
00485 IF LEFT$(T$,1)="-" THEN PRINT " ERROR - NEGATIVE NUMBER"
00490 RETURN
00500 E$=" ERROR IN OFFSET OCTAL NOTATION !":C=1
00505 IF MID$(T$,4,1)="." THEN T$=LEFT$(T$,3)+" "+RIGHT$(T$,3)
00510 FOR I=LEN(T$) TO 1 STEP -1
00520 Y1=VAL(MID$(T$,I,1))
00530 IF MID$(T$,I,1)=" " THEN Y=1:C=C-1
00540 IF Y1>7 THEN PRINT E$
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MINIBUG - HORNE < CONT'D >

```
00550 IF C=3 THEN GOTO 600
00560 IF C=6 THEN GOTO 600
00570 C=C+1
00580 NEXT I
00590 RETURN
00600 IF Y1>3 THEN PRINT E$
00610 GOTO 570
00620 H$=" ":X=16384:T=VAL(T$)
00630 H=INT(T/X):T=T-(H*X)
00640 H=H+48:H$=H$+CHR$(H)
00650 X=X/8
00660 IF X<1 THEN RETURN
00670 IF X=32 THEN H$=H$+" ":X=64
00680 GOTO 630
00690 X=0:T=1:C=0
00700 FOR I=LEN(T$) TO 1 STEP -1
00710 H=ASC(MID$(T$,I,1))-48:C=C+1
00720 IF H=-16 THEN T=256:C=C+1:GOTO 750
00730 IF C=4 THEN T=256
00740 X=X+(H*T):T=T*8
00750 NEXT I
00760 RETURN
00770 C1$=" OFFSET OCTAL NOTATION"
00780 C2$=" DECIMAL NOTATION"
00790 C3$=" STARTING ADDRESS ? - "
00800 C4$=" IS THAT OFFSET OCTAL OR DECIMAL NOTATION ? O/D - "
00810 RETURN
00820 PRINT "          ENTER NUMERALS IN";C2$
00830 PRINT : LINE INPUT " READY - ";T$
00840 B$="D":GOSUB 470
00850 GOSUB 620
00860 PRINT T$;" DECIMAL = ";H$;LEFT$(C1$,13)
00870 LINE INPUT " AGAIN ? Y/N - ";F1$
00880 IF F1$="Y" THEN GOTO 830
00890 GOTO 360
00900 PRINT :PRINT "          ENTER NUMERAL IN";C1$
00910 PRINT :LINE INPUT " READY - ";T$
00920 GOSUB 500
00930 GOSUB 690
00940 PRINT T$;LEFT$(C1$,13);" = ";X;LEFT$(C2$,8)
00950 LINE INPUT " AGAIN ? Y/N - ";F2$
```

PROGRAM NAME: MINIBUG - HORNE < CONT'D >

```
00960 IF F2$="Y" THEN GOTO 910
00970 GOTO 360
00980 PRINT :CNTRL 2,2
00990 LINE INPUT " DO YOU WANT TO DISPLAY A REGISTER OR MEMORY ?
R/M - ";P$
01000 IF P$="R" THEN POKE 8199,2:GOTO 1030
01010 IF P$="M" THEN POKE 8199,1:GOTO 1130
01020 GOTO 360
01030 LINE INPUT " WHICH REGISTERS - AF,BC,DE,HL,PC,SP ? - ";R$
01040 X=8197
01050 IF R$="AF" THEN POKE X,2:GOTO 1120
01060 IF R$="BC" THEN POKE X,4:GOTO 1120
01070 IF R$="DE" THEN POKE X,6:GOTO 1120
01080 IF R$="HL" THEN POKE X,8:GOTO 1120
01090 IF R$="PC" THEN POKE X,10:GOTO 1120
01100 IF R$="SP" THEN POKE X,0:GOTO 1120
01110 GOTO 440
01120 PRINT " IT'S DONE - H8 DISPLAY WILL REMAIN ON . ":GOTO 360
01130 PRINT : LINE INPUT " WHAT ADDRESS ? - ";M$
01140 LINE INPUT C4$;B$
01150 T$=M$:GOSUB 470
01160 IF B$="D" THEN GOTO 1270
01170 T$=M$:GOSUB 690
01180 T$=STR$(X):GOSUB 620
01190 M$=H$
01200 T$=LEFT$(M$,4)
01210 GOSUB 690
01220 POKE 8213,X
01230 T$=RIGHT$(M$,3)
01240 GOSUB 690
01250 POKE 8212,X
01260 GOTO 1120
01270 GOSUB 620
01280 M$=H$
01290 GOTO 1200
01300 PRINT :LINE INPUT C3$;N$
01310 LINE INPUT C4$;B$
01320 T$=N$:GOSUB 470
01330 LINE INPUT " WHICH NOTATION DO YOU WANT DATA LISTED IN ? 0
/D - ";B1$
01340 INPUT " HOW MANY BYTES TO LIST ? - ";N
```



<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: MINIBUG - HORNE < CONT'D >

```
01350 IF B$="O" THEN PRINT " OFFSET OCTAL ADDRESS - ";GOSUB 690
0
01360 IF B$="D" THEN PRINT " DECIMAL ADDRESS - ";
01370 IF B1$="O" THEN PRINT " OCTAL DATA"
01380 IF B1$="D" THEN PRINT " DECIMAL DATA"
01390 IF B$="D" THEN X=VAL(T$):GOTO 1420
01400 IF B$="O" THEN GOTO 1420
01410 PRINT " SORRY ! - ";GOTO 360
01420 C=1
01430 FOR J=X TO X+N-1
01440 IF B$="O" THEN T$=STR$(J):GOSUB 620
01450 IF B$="O" THEN PRINT H$;" - ";
01460 IF B$="D" THEN PRINT J;" - ";
01470 IF B1$="D" THEN PRINT PEEK(J):GOTO 1510
01480 T$=STR$(PEEK(J)):GOSUB 620
01490 Q$=RIGHT$(H$,3)
01500 PRINT Q$
01510 C=C+1:IF C=10 THEN PRINT " PUSH SPACE BAR TO CONTINUE"
01520 IF C=10 THEN PAUSE :C=1
01530 NEXT J
01540 GOTO 360
01550 PRINT :LINE INPUT C3$;T$
01560 LINE INPUT C4$;B$
01570 GOSUB 470
01580 LINE INPUT " WHAT NOTATION WILL DATA BE IN ? O/D = ";B1$
01590 INPUT " HOW MANY BYTES TO ALTER ? - ";N
01600 IF B$="O" THEN PRINT "OFFSET OCTAL ADDRESS - ";GOSUB 690
01610 IF B$="D" THEN PRINT "DECIMAL ADDRESS - ";
01620 IF B1$="O" THEN PRINT "OCTAL DATA":GOTO 1650
01630 IF B1$="D" THEN PRINT "DECIMAL DATA":GOTO 1650
01640 PRINT "SORRY ! - ";GOTO 360
01650 IF B$="D" THEN X=VAL(T$)
01660 IF B$="O" THEN GOSUB 690
01665 IF X<8192 THEN PRINT " THAT'S ROM - CAN'T BE ALTERED":GOTO 360
01670 FOR J=X TO X+N-1
01680 IF B$="O" THEN T$=STR$(J):GOSUB 620
01690 IF B$="O" THEN PRINT H$;" - ";
01700 IF B$="D" THEN PRINT J;" - ";
01710 LINE INPUT " ";T$
01720 IF B1$="D" THEN X=VAL(T$):GOTO 1740
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MINIBUG - HORNE < CONT'D >

```
01730 GOSUB 690
01740 IF X>256 THEN PRINT "ERROR - TOO LARGE ! " : GOTO 360
01750 POKE J,X
01760 NEXT J
01770 GOTO 360
01780 PRINT : PRINT " ANY DATA WILL BE REPLACED BY '0'S ."
01790 LINE INPUT C3$;T$
01800 LINE INPUT C4$;B$
01810 INPUT " HOW MANY BYTES TO CLEAR ? - " : N
01820 GOSUB 470
01830 IF B$="0" THEN GOSUB 690
01840 IF B$="D" THEN X=VAL(T$)
01850 IF X<17455 THEN PRINT " YOU WILL CLEAR PART OF EX.B.H.BASIC
"
01860 IF X<8192 THEN PRINT " NO! - THAT'S PAM-8 ROM (CANT BE ALTE
RED)"
01870 LINE INPUT " DO YOU WISH TO ABORT AT THIS POINT ? Y/N -
" : A7$
01880 IF A7$="N" THEN PRINT " -RUNNING-" : GOTO 1900
01890 GOTO 360
01900 FOR J=X TO X+N-1
01910 POKE J,0
01920 NEXT J
01930 PRINT " IT'S DONE - WHAT NEXT ? " : GOTO 360
01940 LINE INPUT C3$;T$
01950 LINE INPUT C4$;B$
01960 GOSUB 470
01970 IF B$="D" THEN GOSUB 620
01980 IF B$="0" THEN GOSUB 690
01990 IF B$="0" THEN T$=STR$(X) : GOSUB 620
02000 T$=LEFT$(H$,4) : GOSUB 690
02010 POKE 17268,X
02020 T$=RIGHT$(H$,3) : GOSUB 690
02030 POKE 17267,X
02040 LINE INPUT " DOES THE PROGRAM END WITH A 'RETURN' (311) ? Y
/N - " : Y$
02050 IF Y$="Y" GOTO 2080
02060 IF Y$="N" THEN PRINT " CONTROL WILL REVERT TO H8 ! " : GOTO 20
80
02070 GOTO 360
02080 LINE INPUT " DO YOU WISH TO ABORT AT THIS POINT ? Y/N - " : N
$
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CHI-SQUARE - FORSYTHE

```
00010 REM CHI SQUARE FOR R BY C CONTINGENCY TABLE. FORSYTHE 6/78
00020 INPUT "HOW MANY ROWS,COLUMNS ?";R,C
00030 DIM F(R,C),E(R,C)
00040 FOR I=1TO R
00050 PRINT "ENTER THE DATA FOR ROW ";I
00060 FOR J=1TO C
00070 PRINT "COLUMN ";J;
00080 INPUT N
00085 F(I,J)=N
00090 F(0,J)=F(0,J)+N
00100 F(I,0)=F(I,0)+N
00110 T=T+N
00120 NEXT J:NEXT I
00130 F(0,0)=T
00140 U1=T
00150 PRINT
00160 FOR I=1TO R
00180 P=F(I,0)/T
00185 P1=INT(1000*P+5)/10
00187 PRINT "ROW ";I;" IS ";F(I,0);" CASES (";P1;" PERCENT )"
00188 PRINT "      EXPECTED VALUES: ";
00190 FOR J=1TO C
00200 U=F(0,J)*P
00210 X=F(I,J)-U
00215 C2=C2+X*X/U
00220 E(I,J)=U
00230 IF U<U1 THEN U1=U
00240 PRINT U,
00250 NEXT J
00270 PRINT
00280 NEXT I
00290 PRINT :PRINT "CHI SQUARE=";C2;
00295 D=(R-1)*(C-1)
00297 PRINT "DEGREES OF FREEDOM ARE ";D
00300 PRINT "SMALLEST EXPECTED VALUE IS ";U1
00310 STOP
```

.....



PROGRAM NAME: EXPANSION CHAMBER - FULLER

```
00010 REM          PROGRAM 'EXPANSION CHAMBER'
00012 REM          BY: WILLIAM E. FULLER
00014 REM          IN BH BASIC VERSION 01.02.01
00015 REM          CONSOLE LENGTH = 132
00016 REM          DATE: 4/NOV/78
00018 REM
00019 REM
00022 REM          THIS PROGRAM WILL DESIGN TWO STROKE ENGINE EXPANSION
CHAMBERS.
00024 REM          IT WILL DESIGN THESE EXHAUST SYSTEMS FOR ANY PERFORMA
NCE LEVEL
00026 REM          FROM CONSERVATIVE TO VERY HIGH PERFORMANCE COMPETITIO
N ONLY
00028 REM          TYPES. AS THE FABRICATION AND TESTING OF EXPANSION C
HAMBERS
00030 REM          IS VERY TIME CONSUMING AND EXPENSIVE, THIS PROGRAM WI
LL ENABLE
00032 REM          YOU TO TRY OUT DIFFERENT DESIGNS ON PAPER AT VERY LIT
TLE COST
00034 REM          IN TIME OR MONEY. AFTER THE DESIGN IS DECIDED UPON Y
OU HAVE
00036 REM          A DRAWING OF THE CHAMBER WITH ALL NECESSARY DATA TO B
UILD ONE.
00038 REM          WHILE TRYING VARIOUS DESIGNS, YOU MAY ALSO DETERMINE T
HE INLET
00040 REM          TRACT-CRANKCASE RESONANT FREQUENCY AND CENTERLINE LEN
GTHS OF
00042 REM          CURVED PIPE.
00044 REM
00046 REM
00100 LINE INPUT "DESIGNED FOR: NAME ?";N$
00120 PRINT :PRINT
00140 INPUT "ENGINE DISPLACEMENT IN CC'S ?";C
00160 PRINT :PRINT
00180 LINE INPUT "MAKE OF ENGINE ?";N1$
00200 PRINT :PRINT
00220 INPUT "DESIGN FOR PEAK OUTPUT AT WHAT RPM ?";R
00240 PRINT :PRINT
00260 INPUT "HALF OF THE DIFFUSER'S ANGLE OF DIVERGENCE IN DEG. ?
";A1
00265 IF A1<2 THEN PRINT "USEFUL DESIGN LIMITS 2 TO 7.5 DEGREES."
:GOTO 260
```



PROGRAM NAME: EXPANSION CHAMBER - FULLER

```
00260 IF A1>7.5 THEN PRINT "USEFUL DESIGN LIMITS 2 TO 7.5 DEGREES
." :GOTO 260
00280 PRINT :PRINT
00300 INPUT "HALF OF THE BAFFLE CONE'S ANGLE OF CONVERGENCE IN DE
G. ?":A2
00305 IF A2<6 THEN PRINT "USEFUL DESIGN LIMITS 6 TO 13 DEGREES.":
GOTO 300
00308 IF A2>13 THEN PRINT "USEFUL DESIGN LIMITS 6 TO 13 DEGREES."
:GOTO 300
00320 PRINT :PRINT
00340 INPUT "OPEN PERIOD OF EXHAUST IN DEG. ?":E
00360 PRINT :PRINT
00380 INPUT "INLET PIPE INSIDE DIAMETER IN INCHES ?":D1
00400 PRINT :PRINT
00420 PRINT "THE THREE AREAS OF DESIGN ARE:"
00440 PRINT "1=CONSERVATIVE."
00460 PRINT "2=MODERATE."
00480 PRINT "3=VERY HIGH PERFORMANCE."
00500 INPUT "TYPE OF DESIGN 1,2 OR 3 ?":P
00520 PRINT :PRINT
00540 PRINT "DO YOU WISH TO DO INTAKE TRACT-CRANKCASE RESONANCE"
00545 LINE INPUT "CALCULATION (Y/N) ?":B$
00560 IF B$<>"Y" GOTO 600
00565 PRINT :PRINT
00580 INPUT "CROSS-SECTIONAL AREA OF INLET IN SQUARE INCHES ?":Z
00600 PRINT :PRINT
00620 INPUT "INLET PIPE LENGTH IN INCHES ?":W
00640 PRINT :PRINT
00660 INPUT "CRANKCASE VOLUME IN CUBIC INCHES ?":V
00680 PRINT :PRINT
00700 PRINT "WANT CENTERLINE LENGTH OF CURVED PIPE"
00704 LINE INPUT "CALCULATION (Y/N) ?":B1$
00720 IF B1$<>"Y" GOTO 5000
00740 PRINT :PRINT
00760 INPUT "ANGLE OF BEND IN DEGREES ?":R5
00780 PRINT :PRINT
00800 INPUT "RADIUS OF BEND IN INCHES ?":R6
00820 PRINT :PRINT
05000 GOSUB 40000
05050 GOSUB 40100
05100 GOSUB 40300
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: EXPANSION CHAMBER - FULLER < CONT'D >

```

05150 GOSUB 40400
05200 GOSUB 40600
05250 GOSUB 40700
05300 GOSUB 40850
05350 GOSUB 41000
05400 GOSUB 41100
05450 GOSUB 41200
05500 GOSUB 41300
05550 IF B$="Y" THEN GOSUB 41400
05600 IF B1$="Y" THEN GOSUB 41500
06000 LINE INPUT "IS PRINTER READY (Y/N) ?";J$
06020 IF J$<>"Y" GOTO 6000
09890 PRINT :PRINT :PRINT
09900 PRINT "=====
=="
09910 PRINT "***** TWO STROKE ENGINE EXPANSION CHAMBER DESIGN *****
**"
09912 PRINT "      ENGINE MANUFACTURER: ";N1$
09914 PRINT "      ENGINE DISPLACEMENT: ";C;" CC."
09916 PRINT "      DESIGNED FOR: ";N$
09920 PRINT "=====
=="
09930 PRINT :PRINT
09995 PRINT "                                OUTLET"
10000 PRINT "-----X----->*****"
10010 PRINT "      !                *      *"
10020 PRINT "      !                *      *"
10030 PRINT "      !                *      *"
10040 PRINT "      !                *      *      (OU
TLET PIPE AREA)"
10050 PRINT "      (L7)                *      *"
10060 PRINT "      !                *      *"
10070 PRINT "      !                *      *-----<
D3)"
10080 PRINT "      !                *      *"
10090 PRINT "      !                *      *"
10100 PRINT "      !                *      *-----
-----X-----"
10110 PRINT "      !                *      *
      !                *      *
10120 PRINT "      !                *      *
      !                *      *

```

PROGRAM NAME: EXPANSION CHAMBER - FULLER

```

10130 PRINT "-----X----->* - -*
          !"
10140 PRINT "          !          *          *
          !"
10150 PRINT "          !          *          *
          !"
10160 PRINT "          !          *          T          * (BA
FFLE CONE AREA)
10170 PRINT "          !          *          !          *
          !"
10180 PRINT "          (L6)          *          .          *
          (L2)"
10190 PRINT "          !          *          !          *
          !"
10200 PRINT "          !          *          .          *
          !"
10210 PRINT "          !          *          !          *
          !"
10220 PRINT "          !          * ANGLE A2 .
*          !"
10230 PRINT "          !          *<----->!
*          !"
10240 PRINT "          !          *          .
*          !"
10250 PRINT "          !          *          !
*          !"
10260 PRINT "-----X-----X----->* - - - - . - - - -
          !          !          *          !
10270 PRINT "          !          !          *          !
          !"
10280 PRINT "          !          !          *          .
          !"
10290 PRINT "          !          !          *          !
          !"
10300 PRINT "          !          !          *          .
          * (CHAMBER AREA)"
10310 PRINT "          !          (L5)          *          !
          !"
10320 PRINT "          !          !          -->*          .
          *<----->(D2)"
10330 PRINT "          !          !          *          !
          !"

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: EXPANSION CHAMBER - FULLER

```

10340 PRINT "      !      !      *      .
      *"
10350 PRINT "      !      !      *      !
      *"
10360 PRINT "----- ! -----X----->*-- - - - - . - - - -
      -*"
10370 PRINT "      !      !      *      !
      *"
10380 PRINT "      !      !      *      .
      *"
10390 PRINT "      !      !      *      !
      *"
10400 PRINT "      !      !      *  ANGLE A1  .
      *"
10410 PRINT "      !      !      *<----->!
      *"
10420 PRINT "      !      (L4)      *      .
      *"
10430 PRINT "      !      !      *      !
      *"
10440 PRINT "      !      !      *      (LT)
      *"
10450 PRINT "      !      !      *      .      *
      "
10460 PRINT "      !      !      *      !      *
      (DIFFUSER AREA)"
10470 PRINT "      (L1)      !      *      .      *"
10480 PRINT "      !      !      *      !      *"
10490 PRINT "      !      !      *      !      *"
10500 PRINT "      !      !      *      !      *"
10510 PRINT "      !      !      *      !      *"
10520 PRINT "      !      !      *      !      *"
10530 PRINT "      !      !      *      !      *"
10540 PRINT "      !      !      *      !      *"
10550 PRINT "----- ! -----X----->*-- - - - - -*"
10560 PRINT "      !      !      *      !      *"
10570 PRINT "      !      !      *      !      *"
10580 PRINT "      !      !      *      !      *"
10590 PRINT "      !      (L3)      *      !      *"
10600 PRINT "      !      !      *      !      *<-----
      -(D1)"
10610 PRINT "      !      !      *      .      *"

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: EXPANSION CHAMBER - FULLER < CONT'D >

```
10620 PRINT "          !          !          *          !          *"
10630 PRINT "          !          !          *          .          * (IN
LET PIPE AREA)"
10640 PRINT "          !          !          *          !          *"
10650 PRINT "          !          !          *          .          *"
10660 PRINT "          !          !          *          !          *"
10670 PRINT "          !          !          *          .          *"
10680 PRINT "-----X-----X----->*****-*****"
10690 PRINT "                                INLET"
12000 PRINT "=====
12050 PRINT "DIMENSION LIST:"
12100 PRINT "=====
12150 PRINT :PRINT
12160 CNTRL 3.30
12200 PRINT "LENGTHS.", "DIAMETERS.", "ANGLES."
12250 PRINT "=====", "=====
12300 PRINT "L1=";L1;" INCHES.", "D1=";D1;" INCHES.", "A1=";A1;" DE
GREES."
12350 PRINT "L2=";L2;" INCHES.", "D2=";D2;" INCHES.", "A2=";A2;" DE
GREES."
12400 PRINT "L3=";L3;" INCHES.", "D3=";D3;" INCHES."
12450 PRINT "L4=";L4;" INCHES."
12500 PRINT "L5=";L5;" INCHES."
12550 PRINT "L6=";L6;" INCHES."
12600 PRINT "L7=";L7;" INCHES."
12625 CNTRL 3.14
12650 PRINT
12700 PRINT "NOTE 1: TUNED LENGTH IN INCHES =" ;L9
12750 PRINT "NOTE 2: THE DISTANCE FROM THE BAFFLE CONE'S INLET TO
THE"
12800 PRINT "          MEAN POINT OF REFLECTION =" ;L8;" INCHES."
12850 IF B$<>"Y" GOTO 13500
12900 PRINT
12950 PRINT "INTAKE TRACT-CRANKCASE RESONANCE DATA:"
13000 PRINT "=====
13050 PRINT "CRANKCASE VOLUME =" ;V;" CUBIC INCHES."
13100 PRINT "INLET CROSS-SECTIONAL AREA =" ;Z;" SQUARE INCHES."
13150 PRINT "INLET PIPE LENGTH = " ;W;" INCHES."
13155 PRINT
13200 PRINT "NOTE 1: THE ABOVE VALUES WILL RESULT IN A INTAKE TRA
CT-"
13220 PRINT "          CRANKCASE RESONANCE OF " ;V2;" HERTZ/SEC."
```



PROGRAM NAME: EXPANSION CHAMBER - FULLER < CONT'D >

```
13225 PRINT "          OR AT AN ENGINE RPM OF ";U2*60
13500 IF B1#<>"Y" GOTO 35000
13550 PRINT
13600 PRINT "CURVED PIPE DATA:"
13650 PRINT "=====
13700 PRINT "ANGLE OF CURVE IN DEGREES =";R5
13750 PRINT "RADIUS OF CURVE IN INCHES =";R6
13800 PRINT
13850 PRINT "NOTE 1: THE SECTION OF CURVED PIPE WILL HAVE A CENTE
RLINE"
13900 PRINT "          LENGTH OF ";Z3;" INCHES."
14000 PRINT :PRINT :PRINT
35000 END
40000 L9=(1700*E)/R
40050 RETURN
40100 IF P=1 THEN L3=D1*11
40150 IF P=2 THEN L3=D1*8.5
40200 IF P=3 THEN L3=D1*6
40250 RETURN
40300 D2=SQR((6.25*(D1↑2)))
40350 RETURN
40400 IF P=1 THEN D3=D1*.62
40450 IF P=2 THEN D3=D1*.60
40500 IF P=3 THEN D3=D1*.58
40550 RETURN
40600 L7=D3*12
40650 RETURN
40700 I1=1/((TAN(A1/57.296)))
40750 L4=I1*((D2-D1)/2)
40800 RETURN
40850 I2=1/((TAN(A2/57.296)))
40900 L2=I2*(D2/2)
40950 RETURN
41000 L1=L9-(L2/2)
41050 RETURN
41100 L5=L1-(L3+L4)
41150 RETURN
41200 L6=I2*((D2-D3)/2)
41250 RETURN
41300 L8=((D2/2)*I2)/2)
41350 RETURN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: EXPANSION CHAMBER - FULLER < CONT'D >

```
41400 U1=(1100/(2*3.14159))
41450 U2=U1*(SQR(Z/(U*(W+(.5*(SQR(3.14159*Z)))))))
41455 RETURN
41500 Z3=R5*R6*.01745
41550 RETURN
```

.....

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS

```
00010 REM *****
*****
00020 REM          MULTIPLE COLUMN FILE MANAGEMENT PROGRAM
00030 REM *****
*****
00040 REM          WRITTEN BY:  MILTON K. HICKS      9/78
00050 REM                      222 PACIFIC AVENUE
00060 REM                      LONG BEACH, CA 90802
00070 REM
00090 REM          USING:  H8 /EX.BASIC VERSION 10.02.01
00100 REM                  30K MEMORY
00110 REM                  H9 / H36 PRINTER (OPTIONAL)
00120 REM
00130 REM *****
*****
00140 REM          NEW PROGRAM INSTRUCTIONS
00150 REM *****
*****
00160 GOSUB 8000:REM ..CLEAR SCREEN
00170 CNTRL 4,1
00190 LINE INPUT "HAVE YOU WORKED THIS PROGRAM BEFORE? <YES-NO> "
:A$
00200 IF ASC(A$)=78 THEN GOSUB 9000:GOTO 250
00210 IF ASC(A$)=89 THEN GOSUB 10000:GOTO 250
00220 PRINT :PRINT "HUH??  THAT'S NOT A <YES> OR <NO> ANSWER!!"
00230 PRINT "PLEASE RE-ENTER THE CORRECT RESPONSE:" :PRINT :GOTO 1
90
00250 PRINT :PRINT "I WILL NOW DELETE ALL OF THE PROGRAM FROM MY
MEMORY"
00260 PRINT "THAT WE HAVE COMPLETED THUS FAR.":PAUSE 1000
00270 POKE 8302,85:POKE 8303,78:POKE 8304,13:REM ..UNLOCK
00280 POKE 8305,68:POKE 8306,69:POKE 8307,76:REM ..DELETE
00290 POKE 8308,57:POKE 8309,48:POKE 8310,48:POKE 8311,48:REM ..9
000
00300 POKE 8312,44:POKE 8313,49:POKE 8314,49:POKE 8315,48:POKE 83
16,48
00310 POKE 8317,48:POKE 8318,13:REM ..11000 CR
00320 POKE 8319,67:POKE 8320,79:POKE 8321,78:POKE 8322,13:REM ..C
ONTINUE
00330 POKE 8301,21
00340 STOP
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
00400 REM *****
*****
00410 REM                               INITIALIZE NEW PROGRAM
00420 REM *****
*****
00430 GOSUB 8000:REM ..CLEAR SCREEN
00440 LINE INPUT "TITLE OF YOUR NEW PROGRAM? ";T$
00450 PRINT :PRINT :INPUT "NUMBER OF COLUMNS DESIRED? <1-20> ";N
00460 N=INT(N):IF N>0 AND N<21 THEN 500
00470 PRINT :PRINT "HUH?? THAT'S NOT A NUMBER BETWEEN <1> AND <2
0>."
00480 PRINT "PLEASE RE-ENTER YOUR DATA:":GOTO 450
00490 REM
00500 GOSUB 8000:REM ..CLEAR SCREEN
00510 N=N-1:REM ..COMPENSATE FOR USING '0' IN ARRAYS
00520 DIM T$(N),T(N),B$(N),C$(N,100)
00530 S=0:FOR A=0 TO N
00540 PRINT "TITLE FOR COLUMN #":A+1;
00550 LINE INPUT "? ";T$(A)
00560 S=S+LEN(T$(A)):NEXT A
00570 S=S+2*N
00580 GOSUB 8000:REM ..CLEAR SCREEN
00600 PRINT "    YOUR COLUMN TITLES, WITH 2 SPACES BETWEEN THEM, T
AKE UP"
00610 PRINT S;"SPACES. IF THIS EXCEEDS THE SIZE OF YOUR PRINT ZON
E ON"
00620 PRINT "YOUR PRINTER OR MONITOR, YOU WILL HAVE TO RE-ENTER T
HE TITLES"
00630 PRINT "IN ABBREVIATED FORM TO REDUCE THE SPACES USED.":PRIN
T
00640 PRINT :LINE INPUT "IS YOUR SPACING OKAY? <YES-NO> ";A$
00650 IF ASC(A$)=89 THEN GOSUB 8000:GOTO 680
00660 IF ASC(A$)=78 THEN GOSUB 8000:GOTO 530
00670 PRINT :PRINT "<INCORRECT RESPONSE> RE-ENTER":PRINT :GOTO 64
0
00680 PRINT "    OKAY, NOW....BEFORE WE MAKE ANY PERMANENT CHANGES
IN THE"
00690 PRINT "PROGRAM....LET'S ENTER ONE ROW OF DATA FOR YOUR PROG
RAM IN"
00700 PRINT "ORDER TO SET THE TAB SPACING AND PRINT THE RESULTS."
:PRINT
00710 PRINT "    ENTER THE TOTAL NUMBER OF SPACES IN THE PRINT ZON
E YOU"
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS

```
00720 PRINT "WANT TO USE...H9 IS [80] OR LESS. IF YOU HAVE A PRI
NTER."
00730 PRINT "IT WILL BE THE PAPER WIDTH UP TO A MAXIMUM OF [132]"
:PRINT
00740 INPUT "TOTAL SPACES = ";X
00750 X=INT(X):IF X>=5 AND X<=132 THEN 790
00760 PRINT :PRINT "YOU HAVE ENTERED A <WRONG NUMBER> THAT IS EIT
HER TOO"
00770 PRINT "LARGE OR SMALLER THAN YOUR COLUMN TITLE SPACES.":PRI
NT
00780 PRINT "PLEASE RE-ENTER THE INFORMATION.":PRINT :PRINT :GOTO
710
00790 REM
00800 GOSUB 8000:REM ..CLEAR SCREEN
00810 PRINT " YOU HAVE /";X-S;"/SPACES LEFT ON YOUR PRINTING A
REA TO"
00820 PRINT "ALLOW FOR YOUR DATA TO EXTEND BEYOND THE LENGTH OF Y
OUR"
00830 PRINT "COLUMN TITLES."
00840 PRINT " THIS PROGRAM WILL AUTOMATICALLY ADJUST THE TAB S
ETTINGS"
00850 PRINT "OF YOUR COLUMNS IN THE FUTURE, DEPENDING ON THE LENG
TH OF YOUR"
00860 PRINT "DATA ENTERED,SO YOU WILL NEVER HAVE OVERLAPPING COLU
MNS, BUT"
00870 PRINT "YOU CAN RUN OFF THE PRINT ZONE IF YOU ARE NOT CAREFU
L."
00880 PRINT :PRINT "<PAUSE>":PAUSE :PRINT
00890 PRINT "IN ENTERING THE FIRST ROW OF FILE DATA, DETERMIN
E THE"
00900 PRINT "APPROXIMATE 'MAXIMUM' SPACES DATA WILL TAKE UP IN EA
CH COLUMN"
00910 PRINT "AND, AS YOU ENTER THE DATA OF THAT COLUMN, ADD SPACE
S AFTER THE"
00920 PRINT "DATA UP TO THE COLUMN WIDTH YOU WANT."
00930 PRINT " THE INITIAL TAB SETTINGS WILL THEN BE SET.":PRIN
T
00940 PRINT "<PAUSE>":PAUSE :PRINT
00950 GOSUB 8000:REM ..CLEAR SCREEN
00960 PRINT "HERE WE GO. . . ":PRINT :PRINT
00970 S1=S:FOR A=0 TO N
```



<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
00980 PRINT T$(A);:LINE INPUT "? ";C$(A,P)
00990 IF LEN(C$(A,P))<=LEN(T$(A)) THEN T(A)=LEN(T$(A))+2:GOTO 102
0
01000 IF LEN(C$(A,P))>LEN(T$(A)) THEN T(A)=LEN(C$(A,P))+2
01010 S1=S1+T(A)-LEN(T$(A))-2
01020 PRINT TAB(55);X-S1;"SPACES LEFT":NEXT A
01030 PRINT "VERIFICATION OF DATA ENTERED:":PRINT
01040 FOR A=0 TO N:PRINT C$(A,P):NEXT A:PRINT
01050 LINE INPUT " _ IS DATA CORRECT? _ <YES-NO> ";A$
01060 IF ASC(A$)=78 THEN PRINT "## REJECTED ##":GOTO 950
01070 IF ASC(A$)=89 THEN P=P+1:S=S1:GOTO 1100
01080 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 1030
01100 GOSUB 8000:REM ..CLEAR SCREEN
01110 PRINT "      SO FAR SO GOOD.  IF YOU PLAN TO USE A SEPARATE P
RINTER."
01120 PRINT "PLEASE MAKE SURE IT IS TURNED ON AND READY TO GO."
01130 PRINT :PRINT "<PAUSE>";:PAUSE :PRINT
01140 IF Z=0 THEN GOSUB 8000:GOTO 1160
01150 PORT= 254:PRINT
01160 PRINT TAB(S/2-LEN(T$)/2);T$
01170 FOR A=0 TO S:PRINT CHR$(42);:NEXT A:PRINT
01180 FOR A=0 TO N:PRINT T$(A)SPC(T(A)-LEN(T$(A))-1);:NEXT A:PRINT

01190 FOR A=0 TO S:PRINT CHR$(42);:NEXT A:PRINT
01200 PRINT
01210 FOR A=0 TO N:PRINT C$(A,0)SPC(T(A)-LEN(C$(A,0))-1);:NEXT A:P
RINT
01215 IF Z=0 THEN PRINT :GOTO 1230
01220 PORT= 250:PRINT :REM ..RETURN TO COMMAND MONITOR
01230 LINE INPUT "IS THIS PRINTOUT WHAT YOU WANT? <YES-NO> ";A$
01240 IF ASC(A$)=89 THEN 1300
01250 IF ASC(A$)=78 THEN 1270
01260 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 1230
01270 PRINT :PRINT "OKAY.  THE BEST THING TO DO IS START OVER FRO
M THE"
01280 PRINT "THE BEGINNING.":CLEAR :GOTO 430
01300 GOSUB 8000:REM ..CLEAR SCREEN
01310 PRINT "      <THANK YOU>  I AM PLEASED IT WORKED OUT.  PLEASE
REMEMBER"
01320 PRINT "THAT THE TAB SETTINGS ARE [NOT] PERMANENT, BUT WILL
CHANGE IF"
01330 PRINT "YOUR DATA ENTERED HAPPENS TO BE LONGER THAN WHAT HAS
BEEN SET"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS

```
01340 PRINT "UP HERE...SO BE CAREFUL.":PRINT
01350 PRINT "      I WILL NOW DELETE THE EXTRANEIOUS PROGRAM MATERIA
L WE HAVE"
01360 PRINT "GONE THROUGH. [ DATA WILL BE RETAINED ]":PRINT
01370 PRINT "<PAUSE>";:PAUSE :PRINT
01380 POKE 8302,68:POKE 8303,69:POKE 8304,76:REM ..DELETE
01390 POKE 8305,52:POKE 8306,48:POKE 8307,44:REM ..40,
01400 POKE 8308,49:POKE 8309,52:POKE 8310,57:POKE 8311,48:REM ..1
490
01410 POKE 8312,13
01420 POKE 8313,71:POKE 8314,79:POKE 8315,84:REM ..GOTO
01430 POKE 8316,49:POKE 8317,53:POKE 8318,48:POKE 8319,48:REM ..1
500
01440 POKE 8320,13
01450 POKE 8321,67:POKE 8322,79:POKE 8323,78:REM ..CONTINUE
01460 POKE 8324,13:POKE 8301,23
01470 STOP
01500 GOSUB 8000:REM ..CLEAR SCREEN
01510 PRINT "*****"
*****
01520 PRINT "                      COMMAND OPTIONS"
01530 PRINT "*****"
*****
01535 PRINT
01540 PRINT "[1] ADD DATA                      [6] 'GET' DATA FROM TAPE
"
01550 PRINT "[2] DELETE DATA                      [7] 'PUT' DATA TO TAPE"
01560 PRINT "[3] SORT DATA"
01570 PRINT "[4] FULL PRINTOUT"
01580 PRINT "[5] PARTIAL PRINTOUT"
01600 PRINT :INPUT "OPTION NUMBER? ":A
01610 A=INT(A):IF A>0 AND A<8 THEN 1630
01620 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":GOTO 1530
01630 ON A GOTO 1700,1900,2200,3000,3200,4000,4500
01700 GOSUB 8000:REM ..CLEAR SCREEN
01710 PRINT "*****"
*****
01720 PRINT "                      ADD DATA TO FILE"
01730 PRINT "*****"
*****
01740 PRINT
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
01750 PRINT "TO EXIT....ENTER /STOP/":PRINT
01760 FOR A=0 TO N:PRINT T$(A);
01770 LINE INPUT "? ";C$(A,P):IF LEN(C$(A,P))<1 THEN 1770
01780 IF C$(A,P)="STOP" THEN 1500:REM ..RETURN TO COMMAND
01790 PRINT :NEXT A:PRINT
01800 FOR A=0 TO N:PRINT C$(A,P):NEXT A:PRINT
01810 LINE INPUT "__IS DATA CORRECT?__<YES-NO> ";A$
01820 IF ASC(A$)=78 THEN PRINT "## REJECTED ##":GOTO 1740
01830 IF ASC(A$)=89 THEN 1850
01840 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 1800
01850 FOR A=0 TO N
01860 IF LEN(C$(A,P))>T(A)-2 THEN S=S+LEN(C$(A,P))+2-T(A):GOTO 18
80
01870 NEXT A:GOTO 1890
01880 T(A)=LEN(C$(A,P))+2:GOTO 1870
01890 P=P+1:GOSUB 8000:GOTO 1750
01900 GOSUB 8000:REM ..CLEAR SCREEN
01910 PRINT "*****
*****"
01920 PRINT "                                DELETE DATA FROM FILE"
01930 PRINT "*****
*****"
01940 PRINT
01950 PRINT "TO EXIT....ENTER /STOP/":PRINT
01960 PRINT "ENTER THE DATA IN COLUMN /1/ TO INDICATE WHICH ROW"
01970 PRINT "YOU WANT DELETED";
01980 LINE INPUT "? ";A$:IF LEN(A$)<1 THEN 1980
01990 IF A$="STOP" THEN 1500
02000 FOR A=0 TO P-1
02010 IF A$<>C$(0,A) THEN 2100
02020 REM ...FOUND IT...NOW ADJUST LIST
02030 FOR B=A TO P-2
02040 FOR C=0 TO N
02050 C$(C,B)=C$(C,B+1)
02060 NEXT C:NEXT B
02070 P=P-1
02080 GOSUB 8000
02090 GOTO 1950
02100 NEXT A
02110 GOSUB 8000:GOTO 1950
02200 GOSUB 8000:REM ..CLEAR SCREEN
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
02210 PRINT "*****
*****"
02220 PRINT "                      SORT DATA IN FILE"
02230 PRINT "*****
*****"
02240 PRINT
02250 FOR A=0 TO N:PRINT "[ ";MID$(STR$(A+1),2,1);"] ";T$(A)
02260 NEXT A:PRINT
02270 INPUT "WHICH COLUMN NUMBER DO YOU WANT SORTED? ";D
02280 D=INT(D-1):IF D>=0 AND D<=N THEN 2300
02290 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 2270
02300 PRINT :LINE INPUT "IS COLUMN ALL NUMBERS? <YES-NO> ";A$
02310 IF ASC(A$)=78 THEN A1=0:GOTO 2340
02320 IF ASC(A$)=89 THEN A1=1:GOTO 2340
02330 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 2300
02340 CNTRL 4,0
02350 M=P
02360 M=INT(M/2):IF M=0 THEN CNTRL 4,1:GOTO 1500
02370 J=0:K=(P-1)-M
02380 I=J
02390 L=I+M
02400 IF A1=0 THEN IF C$(D,I)<=C$(D,L) GOTO 2480
02410 IF A1=1 THEN IF VAL(C$(D,I))<=VAL(C$(D,L)) GOTO 2480
02420 FOR A=0 TO N:B$(A)=C$(A,L):NEXT A
02430 FOR A=0 TO N:C$(A,L)=C$(A,I):NEXT A
02440 FOR A=0 TO N:C$(A,I)=B$(A):NEXT A
02450 I=I-M
02460 IF I<0 THEN 2480
02470 GOTO 2390
02480 J=J+1
02490 IF J>K THEN 2360
02500 GOTO 2380
03000 GOSUB 8000:REM ..CLEAR SCREEN
03010 PRINT "*****
*****"
03020 PRINT "                      FULL PROGRAM PRINTOUT"
03030 PRINT "*****
*****"
03040 PRINT :PAUSE 500
03050 IF Z=0 THEN GOSUB 8000:GOTO 3080
03060 PRINT "MAKE SURE PRINTER IS READY...<PAUSE>";:PAUSE :PRINT
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
03070 PORT= 254:PRINT
03080 PRINT TAB(S/2-LEN(T$)/2);T$
03090 FOR A=0TO S:PRINT CHR$(42);:NEXT A:PRINT
03100 FOR A=0TO N:PRINT T$(A)SPC(T(A)-LEN(T$(A))-1);:NEXT A:PRINT

03110 FOR A=0TO S:PRINT CHR$(42);:NEXT A:PRINT
03120 PRINT
03130 FOR J=0TO P-1
03140 FOR A=0TO N
03150 PRINT C$(A,J)SPC(T(A)-LEN(C$(A,J))-1);:NEXT A:PRINT
03160 NEXT J
03170 FOR A=0TO S:PRINT CHR$(42);:NEXT A:PRINT
03175 IF Z=0 THEN PRINT "<PAUSE>";:PAUSE :GOTO 1500
03180 PRINT :PORT= 250:GOTO 1500
03200 GOSUB 8000:REM ..CLEAR SCREEN
03210 PRINT "*****"
03220 PRINT "                                PARTIAL PRINTOUT OF DATA"
03230 PRINT "*****"
03235 PRINT
03240 PRINT "TO EXIT....ENTER 'STOP'":PRINT
03260 FOR A=0TO N
03270 PRINT "[ ";MID$(STR$(A+1),2,1); "]" ";T$(A)
03280 NEXT A:PRINT
03300 PRINT "WHICH COLUMN DO YOU WANT TO USE TO SPECIFY LIMITS"
03310 LINE INPUT "FOR A PARTIAL PRINTOUT? ";A$
03320 IF A$="STOP" THEN 1500
03330 IF VAL(A$)>0 THEN L=VAL(A$):GOTO 3350
03340 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 3300
03350 IF L>0 AND L<N+2 THEN L=L-1:GOTO 3370
03360 GOTO 3340
03370 GOSUB 8000:REM ..CLEAR SCREEN
03380 PRINT "ENTER THE BEGINNING AND ENDING DATA IN COLUMN #"L+1
03390 PRINT "TO SPECIFY THE PRINT LIMITS....ENCLOSE IN QUOTES...."
03400 PRINT :INPUT "PRINT LIMITS? <SEPARATE BY COMMA> ";L$,L1$
03410 GOSUB 8000:REM ..CLEAR SCREEN
03440 PRINT "MAKE SURE PRINTER IS READY...<PAUSE>";:PAUSE :PRINT
03445 IF Z=0 THEN GOSUB 8000:GOTO 3460
03450 PORT= 254:PRINT
03460 PRINT TAB(S/2-LEN(T$)/2);T$
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
03470 FOR A=0TO S:PRINT CHR$(42);:NEXT A:PRINT
03480 FOR A=0TO N:PRINT T$(A)SPC(T(A)-LEN(T$(A))-1);:NEXT A:PRINT

03490 FOR A=0TO S:PRINT CHR$(42);:NEXT A:PRINT
03500 PRINT
03510 FOR J=0TO P-1
03520 IF C$(L,J)>=L$ AND C$(L,J)<=L1$ THEN 3550
03530 NEXT J:PRINT
03535 IF Z=0 THEN PRINT "<PAUSE>";:PAUSE :GOTO 1500
03540 PORT= 250:GOTO 1500
03550 FOR A=0TO N
03560 PRINT C$(A,J)SPC(T(A)-LEN(C$(A,J))-1);:NEXT A:PRINT
03570 NEXT J
03580 FOR A=0TO S:PRINT CHR$(42);:NEXT A:PRINT
03590 GOTO 3540
04000 GOSUB 8000:REM ..CLEAR SCREEN
04010 PRINT "*****"
04020 PRINT " 'GET' DATA FROM TAPE"
04030 PRINT "*****"
04040 PRINT
04050 LINE INPUT "IS THE TAPE PLAYER READY? <YES-NO> ";A$
04060 IF ASC(A$)=78 THEN 1500
04070 IF ASC(A$)=89 THEN 4090
04080 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 4040
04090 PRINT :LINE INPUT "TITLE OF FILE WANTED? ";Z$
04100 POKE 8302,85:POKE 8303,78:POKE 8304,13
04110 POKE 8305,71:POKE 8306,69:POKE 8307,90:POKE 8308,36:POKE 83
04120 POKE 8310,89
04130 POKE 8311,67:POKE 8312,79:POKE 8313,78:POKE 8314,13
04140 POKE 8301,13
04150 STOP :GOTO 1500
04500 GOSUB 8000:REM ..CLEAR SCREEN
04510 PRINT "*****"
04520 PRINT " 'PUT' DATA ON TAPE"
04530 PRINT "*****"
04540 PRINT
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
04550 LINE INPUT "IS BLANK TAPE READY? <YES-NO> ";A#
04560 IF ASC(A#)=78 THEN 1500
04570 IF ASC(A#)=89 THEN 4590
04580 PRINT :PRINT "<IMPROPER ENTRY> RE-ENTER":PRINT :GOTO 4540
04590 PRINT :LINE INPUT "TITLE FOR FILE TAPE? ";Z#
04600 POKE 8302,80:POKE 8303,85:POKE 8304,90:POKE 8305,36:POKE 83
06,13
04610 POKE 8307,67:POKE 8308,79:POKE 8309,78:POKE 8310,13
04620 POKE 8301,9
04630 STOP :GOTO 1500
07999 STOP
08000 REM *****
08010 REM                                     CLEAR SCREEN SUBROUTINE
08020 REM *****
08030 FOR A=1 TO 12:PRINT CHR$(10);:NEXT A:PRINT :RETURN
08100 REM *****
09000 GOSUB 8000:REM ..CLEAR SCREEN
09010 PRINT "*****"
09020 PRINT "                                     PROGRAM EDITORIAL"
09030 PRINT "*****"
09040 PRINT
09050 PRINT "      IF YOU HAVE HAD THE PREVIOUS DESIRE TO CREATE A
DATA"
09060 PRINT "FILE PROGRAM TO KEEP A RECORD OF ALL YOUR CHECKS, TE
LEPHONE"
09070 PRINT "NUMBERS, OR A HUNDRED OTHER TYPES OF INFORMATION....
..ONLY"
09080 PRINT "TO FIND THAT EXISTING 'FILE MANAGEMENT' PROGRAMS FOR
SUCH A"
09090 PRINT "PURPOSE RESTRICTED YOU TO THE NUMBER OF COLUMNS AVAI
LABLE...."
09100 PRINT
09110 PRINT "      YOU MAY APPRECIATE THIS PROGRAM, WHICH ORIGINAT
ED FROM"
09120 PRINT "SUCH FRUSTRATIONS."
09130 PRINT :PRINT "<PAUSE>";:PAUSE :PRINT :GOSUB 8000
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS < CONT'D >

```
09140 PRINT "      THIS PROGRAM IS DESIGNED SO THAT THE USER SELEC
TS THE"
09150 PRINT "NUMBER OF COLUMNS DESIRED <1-20> FOR A PARTICULAR DA
TA FILE"
09160 PRINT "AND THEN FURTHER DESIGNATE THE TITLE OF THE FILE AND
THE "
09170 PRINT "INDIVIDUAL COLUMN TITLES. ALL DATA CONTAINED IN A C
OLUMN"
09180 PRINT "IS TREATED AS 'STRING' DATA, SO THE USER HAS FULL FL
EXIBILITY"
09190 PRINT "FOR ASSIGNING 'NUMBER' OR 'STRING' DATA TO A CERTAIN
COLUMN."
09200 PRINT
09210 PRINT "      THIS PROGRAM CAN ALSO BE USED AS A DESIGN TOOL
FOR A"
09220 PRINT "LARGE PROGRAM IN ORDER TO OBTAIN A DESIRED PRINT FOR
MAT BEFORE"
09230 PRINT "WRITING AN INDIVIDUAL PROGRAM FROM SCRATCH."
09240 PRINT "<PAUSE>";:PAUSE :PRINT :GOSUB 8000
09250 PRINT "      SINCE THE PROGRAM IS USED TO CREATE ALL YOUR DA
TA"
09260 PRINT "FILES, WHETHER 2-COLUMN OR 15-COLUMN, YOU WILL HAVE
ONE"
09270 PRINT "'MASTER' PROGRAM TO MANIPULATE THESE DATA FILES AND
CAN"
09280 PRINT "MAKE EXTENSIVE USE OF THE 'PUT' AND 'GET' COMMANDS O
F THE"
09290 PRINT "COMPUTER, RATHER THAN THE 'FLOAD' AND 'FDUMP' THAT W
OULD"
09300 PRINT "BE OTHERWISE NECESSARY IF YOUR PROGRAMS WERE NOT THE
SAME."
09310 PRINT
09320 PRINT "      THE PROGRAM IS DESIGNED TO HELP YOU CREATE A DA
TA FILE"
09330 PRINT "FROM SCRATCH ON A STEP-BY-STEP BASIS, HOWEVER, YOU M
AY HAVE"
09340 PRINT "RUN THE PROGRAM A COUPLE TIMES AT FIRST TO GET THE H
ANG OF"
09350 PRINT "IT."
09360 PRINT "<PAUSE>";:PAUSE :PRINT :GOSUB 8000
09370 PRINT "      <WARNING> ONCE YOU HAVE GONE THROUGH THE INSTR
UCTIONS"
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MULTIPLE FILE MANAGEMENT - HICKS

```
09380 PRINT "AND HAVE DESIGNED YOUR DATA FILE REQUIREMENTS, THESE
PARTS"
09390 PRINT "OF THE PROGRAM ARE NO LONGER REQUIRED AND JUST TAKE
MEMORY"
09400 PRINT "SPACE YOU COULD USE FOR DATA. THEREFORE...."
09410 PRINT
09420 PRINT "                ## THIS PROGRAM SELF-DESTRUCTS ##"
09430 PRINT
09440 PRINT "TO DELETE THESE PROGRAM PORTIONS, SO MAKE SURE YOU H
AVE A"
09450 PRINT "COPY OF THIS 'ORIGINAL' PROGRAM BEFORE YOU GO ON !!!
"
09460 PRINT "<PAUSE>";:PAUSE :PRINT :GOSUB 8000
09470 PRINT "        WHEN YOU HAVE FINISHED CREATING YOUR FIRST DATA
FILE,"
09480 PRINT "YOU SHOULD 'PUT' IT ON TO A BLANK TAPE TO SAVE IT.
THE"
09490 PRINT "PROGRAM HAS A COMMAND FOR THIS SEQUENCE."
09500 PRINT "        YOU SHOULD THEN STOP THE PROGRAM AND 'DUMP' IT
ON TO"
09510 PRINT "A SECOND BLANK TAPE. THIS WILL BECOME YOUR 'MASTER'
PROGRAM"
09520 PRINT "FOR HANDLING YOUR DATA FILES."
09530 PRINT "        YOU WILL THEN HAVE THREE(3) TAPES...(1) THE 'OR
IGINAL'"
09540 PRINT "PROGRAM, (2) THE 'MASTER' PROGRAM, AND (3) YOUR DATA
FILE."
09550 PRINT
09560 PRINT "<PAUSE>";:PAUSE :PRINT :GOSUB 8000
09570 PRINT "        REMEMBER, THE 'ORIGINAL' PROGRAM SELF-DESTRUCTS
AS"
09580 PRINT "YOU CREATE A NEW DATA FILE ROUTINE, SO THE 'ORIGINAL
' MUST"
09590 PRINT "BE USED WHENEVER YOU CREATE A NEW DATA FILE."
09600 PRINT :PRINT
09610 PRINT "        OKAY, LET'S GET ON WITH THE PROGRAM."
09620 PRINT :PRINT "<PAUSE>";:PAUSE :PRINT :GOSUB 8000
10000 GOSUB 8000:REM ..CLEAR SCREEN
10010 PRINT "*****"
*****"
10020 PRINT "                PRINTER VS. MONITOR"
```



## MULTIPLE FILE MANAGEMENT

Hicks

### AUTHORS NOTES

The program is "file management" oriented for the purpose of allowing the user to create his own data files, such as a cancelled check register, name and telephone/address directory, customer and account file, or a hundred other uses that I leave up to the user's imagination and personal needs.

Unlike other such file management programs I have come across, this program is not restricted to the number of columns available for the user's data information. A choice is available in the program run from just 1 column, all the way up to 20 columns, or more (with a simple modification of a program line).

WARNING! THIS PROGRAM SELF-DESTRUCTS WHEN RUN!!

Once the user has completed the program instructions, these portions of the written program become useless and only take up valuable memory space that could otherwise be used for data by the user. Therefore, thanks to Robert Behar and Neal Rogers (REMark Issue #3), I have not only incorporated the 'Put' and 'Get' into the RUNning program, I have included the 'Delete' command.

The program is designed to free the user of the burden and necessity to modify, rewrite, and de-bug, a program in its entirety just because an existing program is restricted to 4 columns and he would like 5 columns, 8 columns, or whatever.

The "master" program contains no such restrictions since the variable specifications are contained in the individually created DATA files. The "original" program is only required when the user wants to create a new data file with different column requirements, title, and column names. Once a new data file is created, the remaining program is always a "master" program that is compatible with any previous data files created.

The program is further designed so that all data entered is "string" data since the program is, after all, a "file" and has no number-crunching requirements. The exclusive use of "string" data consumes massive amounts of memory, especially on large data file programs, but cannot be avoided without creating substantial restriction on the user by designating some columns as "numbers only" to conserve string and memory space.

Due to the ease in which a user can create a data file and its related column headings, this program could possibly be used as a "creative tool" to design the print format of a large program before an individual program is written in final form.



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: BOLT HOLE LOCATER - WRIGHT

```

00001 REM *BOLT HOLE LOCATIONS. WRITTEN BY K.D. WRIGHT, 8-22-78*
00010 PRINT "THIS PROGRAM CALCULATES THE BOLT HOLE LOCATIONS OF A
  GIVEN QUANTITY"
00020 PRINT "OF BOLTS EQUALLY SPACED AROUND THE CIRCUMFERENCE OF
A CIRCLE."
00030 C=3.1416/180:PRINT
00040 INPUT "PLEASE ENTER RADIUS OF BOLT CIRCLE. (IN INCHES) ";R
00050 INPUT "NEXT, ENTER BOLT HOLE QUANTITY ";Q
00060 X=360/Q:PRINT :PRINT "HOLE NUMBER","DEGREES","X AXIS","Y AX
IS":PRINT
00070 FOR L=1TO Q:Z=Z+X:PRINT TAB(3)L,Z,:Y=C*Z:A=SIN(Y)*R:B=COS(Y
)*R
00080 IF ABS(A)<.0001THEN PRINT " 0":GOTO 100
00090 PRINT A,
00100 IF ABS(B)<.0001THEN PRINT " 0":GOTO 120
00110 PRINT B
00120 NEXT
00130 STOP

```

10	17	29	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CALENDER - WILKINSON

```
00001 REM APPOINTMENT/CALENDAR VER.1.0 NOV.7,1978
00002 REM BY WILLIAM A. WILKINSON
00003 REM HEATHKIT ELECTRONIC CENTER
00004 REM 9207 MAPLE STREET
00005 REM OMAHA, NEBRASKA 68134
00006 REM (402) 391-2071
00007 REM UTILIZING THE 'GET' AND 'PUT' ROUTINES BY ROBERT BEHAR
AND NEAL ROGERS
00008 REM 'REMARK' ISSUE 3,1978 PAGE 17 (LINES 65000-65140)
00009 REM THIS PROGRAM CAN RUN IN 16K BY REMOVING ALL THE REMARK
STATEMENTS
00010 REM ESSENTIAL DATA
00012 CNTRL 4,0
00015 DIM D$(6),M$(11),M(11),P$(99)
00016 X$="TYPE THE NUMBER CORRESPONDING TO THE DESIRED FUNCTION:
"
00017 X2$="INCORRECT DATE"
00020 D$(0)="SUN":D$(1)="MON":D$(2)="TUE":D$(3)="WED":D$(4)="THU"
:D$(5)="FRI":D$(6)="SAT"
00025 PRINT :PRINT :PRINT :PRINT TAB(20)"APPOINTMENT/CALENDAR VER
.1.0":PRINT :PRINT
00030 GOTO 2000:REM TO MAIN LOOP
00049 REM FORMAT FOR COMMON TERMINALS
00050 PRINT :PRINT TAB(20)"FORMAT":PRINT
00055 PRINT "(1) TELETYPE OR LINE PRINTER"
00060 PRINT "(2) VIDEO TERMINAL"
00070 PRINT :INPUT X$:T:PRINT :IF T<1OR T>2THEN 50
00075 IF A=2THEN F2=0:REM LOCKS OUT APPOINTMENTS
00080 GOTO 710
00100 REM GET MONTH & YEAR
00105 LINE INPUT "WHICH MONTH? ";M$
00106 IF ASC(M$)<65OR ASC(M$)>83THEN PRINT "THE NAME OF THE MONTH
.":GOTO 105
00110 INPUT "WHAT YEAR? ";Y
00115 IF Y<1753THEN PRINT "THE YEAR MUST BE GREATER THAN 1753 AD.
":GOTO 110
00120 REM CONVERT MONTHS TO NUMBERS
00125 FOR N=0TO 11
00130 READ M$(N),M(N)
00135 NEXT N:RESTORE
00140 FOR N=0TO 11
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CALENDER - WILKINSON < CONT'D >

```
00145 IF LEFT$(M$,3)=LEFT$(M$(N),3)THEN 155
00150 NEXT N
00152 GOTO 105
00155 M=N+1:REM USED IN SUBROUTINE AT LINE 510
00157 GOSUB 545:REM CHECK FOR LEAP YEAR
00160 PRINT :PRINT
00200 IF T<>1THEN 405
00210 PRINT "POSITION PAPER, HIT 'RETURN' WHEN READY."
00220 PRINT CHR$(7):CNTRL 4,1:PAUSE :CNTRL 4,0
00400 REM DRAW CALENDAR
00405 PRINT TAB(20)M$(N);SPC(3)Y
00410 FOR X=0TO 62:PRINT "*";:NEXT X
00415 PRINT :CNTRL 3,10
00420 FOR X=0TO 6:PRINT D$(X);:NEXT X
00425 FOR X=0TO 62:PRINT "*";:NEXT X
00430 PRINT :CNTRL 3,14
00435 GOSUB 510:REM FIND BEGINNING OF MONTH
00437 F4=1:IF F2=1THEN 4060:REM PRINT "*" 'S
00438 REM CORRECT FOR MONTHS BEGINNING ON SUNDAY
00439 IF Z7=0THEN PRINT D;:GOTO 445
00440 PRINT SPC((Z7*10)-1)D;:GOTO 445
00442 IF D>10THEN S=5:GOTO 444
00443 S=6
00444 PRINT SPC(S)D;
00445 D=D+1:IF D>M(N)AND F=1AND F2=1THEN GOSUB 4020:REM CAL & APP
LISTING
00446 IF D>M(N)AND F=1THEN PRINT :PRINT CHR$(7):CNTRL 4,1:PAUSE :
CNTRL 4,0:GOSUB 635:RETURN
00447 IF D>M(N)THEN PRINT :PRINT CHR$(7):CNTRL 4,1:PAUSE :CNTRL 4
,0:RETURN
00450 IF POS(0)>62THEN PRINT :GOSUB 610:GOTO 445
00455 IF F2<>1THEN 442
00457 GOTO 4060
00500 REM COMPUTE THE DAY OF THE WEEK
00510 Z=INT(.6+(1/M)):Z0=Y-Z:Z1=M+12*Z:Z2=Z0/100:Z3=INT(Z2/4):Z4=
INT(Z2)
00520 Z5=INT((5*Z0)/4):Z6=INT(13*(Z1+1)/5):Z7=Z6+Z5-Z4+Z3+D-1
00530 Z7=Z7-(7*INT(Z7/7))
00535 RETURN
00540 REM CHECK FOR LEAP YEAR & TURN OF CENTURY
00545 Y1=4:IF Y/100=INT(Y/100)THEN Y1=400
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CALENDER - WILKINSON < CONT'D >

```
00550 IF Y/Y1=INT(Y/Y1)THEN M(1)=29
00555 RETURN
00600 REM FORMATS CALENDAR TO USER'S TERMINAL
00610 IF F2=1THEN F5=1:GOSUB 4080
00611 IF F6=1THEN 618
00612 F5=0:F6=0
00613 IF T=1THEN PRINT :PRINT :PRINT :PRINT D$:RETURN
00616 PRINT D$:RETURN
00618 F5=0:F6=0
00619 D=VAL(D$)+1:IF D<10THEN D$=LEFT$(STR$(D),2)+"*":GOTO 621
00620 D$=LEFT$(STR$(D),3)+"*"
00621 IF T=1THEN PRINT :PRINT :PRINT :PRINT D$:RETURN
00624 PRINT D$:RETURN
00634 REM LOOP TO PRINT FOR YEAR
00635 IF N=11THEN RETURN :REM END OF YEAR
00637 D=1:REM RESET THE STARTING DATE
00640 N=N+1:M=N+1:PRINT :PRINT
00645 GOSUB 405:REM IF THIS WORKS I'LL FREAK!!
00650 GOTO 535:REM A SOFTWARE MOBIUS STRIP
00700 REM SELECTS PRINTING OF ONE MONTH OR FULL YEAR
00710 LINE INPUT "DO YOU WANT A SINGLE MONTH OR A YEAR? ";A$
00715 D=1:REM SET STARTING DATE
00720 IF LEFT$(A$,1)="M"THEN F=0:GOSUB 105:GOTO 750
00730 IF LEFT$(A$,1)<>"Y"THEN 710
00740 F=1:PRINT "STARTING WITH ":GOSUB 105
00750 RETURN :REM RETURN TO MAIN LOOP
01000 DATA "JANUARY",31,"FEBRUARY",28,"MARCH",31,"APRIL",30,"MAY",
,31
01010 DATA "JUNE",30,"JULY",31,"AUGUST",31,"SEPTEMBER",30,"OCTOBE
R",31
01020 DATA "NOVEMBER",30,"DECEMBER",31
01995 REM MAIN LOOP
02000 PRINT :PRINT :REM RESERVED
02010 PRINT :PRINT TAB(20)"FUNCTIONS":PRINT
02020 PRINT "(1) APPOINTMENTS"
02030 PRINT "(2) PRINT CALENDAR"
02040 PRINT "(3) PRINT CALENDAR WITH APPOINTMENTS"
02050 PRINT :INPUT X$:A
02060 IF A=0OR A>3THEN 2010
02080 ON A GOSUB 3010,50,4010
02090 GOTO 2010
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CALENDER - WILKINSON < CONT'D >

```
03000 REM APPOINTMENTS ROUTINES
03010 PRINT :PRINT TAB(20)"APPOINTMENTS":PRINT
03020 PRINT "(1) LIST APPOINTMENTS"
03030 PRINT "(2) ADD APPOINTMENTS"
03040 PRINT "(3) DELETE APPOINTMENTS"
03042 PRINT "(4) SAVE LIST ON TAPE"
03044 PRINT "(5) GET NEW LIST FROM TAPE"
03046 PRINT "(6) EXIT"
03050 PRINT :INPUT X$:A
03070 ON A GOSUB 3205,3105,3305,3505,3605
03075 IF A=6 THEN RETURN
03080 GOTO 3010
03100 REM ADD APPOINTMENTS
03105 PRINT "TYPE '99,9,9' AFTER 'DATE' WHEN FINISHED."
03115 FOR X=P TO 99
03120 INPUT "DATE? (NOV.9,1978=11,9,1978) ";P1,P2,P3
03122 IF P1=99 THEN P$(X)=STR$(P1):P=X:RETURN
03125 Y=P3:M(1)=28:GOSUB 545:REM CHECK FOR LEAP YEAR
03127 REM CHECK FOR ENTRY ERRORS
03130 IF M(1)=29 AND P1=2 AND P2>29 THEN PRINT X2$:GOTO 3120
03135 IF M(1)<>29 AND P1=2 AND P2>28 THEN PRINT X2$:GOTO 3120
03140 IF P2>30 AND (P1=4 OR P1=6 OR P1=9 OR P1=11) THEN PRINT X2$:GOTO
3120
03145 IF P2<10 OR P2>31 THEN PRINT X2$:GOTO 3120
03150 IF P3<1753 THEN PRINT X2$:GOTO 3120
03155 LINE INPUT "MEMO: ";P$
03157 REM FORMAT MEMO
03158 IF P1<10 THEN P1$=" "+STR$(P1):GOTO 3162
03160 P1$=STR$(P1)
03162 IF P2<10 THEN P2$=" "+STR$(P2):GOTO 3166
03164 P2$=STR$(P2)
03166 P$(X)=P1$+P2$+STR$(P3)+P$
03168 NEXT X:P=X:REM P=100 INDICATES FULL APPOINTMENT BUFFER
03170 RETURN
03200 REM LIST APPOINTMENTS
03205 FOR X=0 TO 99
03210 IF LEFT$(P$(X),3)=" 99" THEN RETURN
03215 IF P$(X)<>"" THEN PRINT P$(X)
03220 NEXT X
03225 RETURN
03300 REM DELETE APPOINTMENTS
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CALENDER - WILKINSON < CONT'D >

```
03305 INPUT "DATE TO BE DELETED (APRIL 1,1978=4,1,1978) ";P1,P2,P
3
03309 REM FORMAT FOR EASY SEARCH
03310 IF P1<10THEN P1$=" "+STR$(P1):GOTO 3320
03315 P1$=STR$(P1)
03320 IF P2<10THEN P2$=" "+STR$(P2):GOTO 3330
03325 P2$=STR$(P2)
03330 P$=P1$+P2$+STR$(P3)
03335 FOR X=0TO 99
03337 F1=0
03339 IF LEFT$(P$(X),3)=" 99"THEN 3355
03340 IF LEFT$(P$(X),14)=P$THEN F1=1:GOSUB 3400
03341 IF F1=1AND LEFT$(A$,1)="N"THEN 3345
03342 IF F1=1THEN 3365
03345 NEXT X
03355 LINE INPUT "DATE NOT FOUND. DO YOU WANT A LISTING? ";A$
03360 IF LEFT$(A$,1)="Y"THEN GOSUB 3205
03365 LINE INPUT "ANY MORE DELETIONS? ";A$
03370 IF LEFT$(A$,1)="Y"THEN PRINT :GOTO 3305
03375 RETURN
03395 REM LAST CHANCE
03400 PRINT "FOUND: ";P$(X)
03405 LINE INPUT "IS THIS THE ONE YOU WANT DELETED? ";A$
03410 IF LEFT$(A$,1)="N"THEN RETURN
03415 REM DELETE FILE, SHUFFLE OTHERS DOWN QUEUE
03420 P=P-1:P$(X)=" "
03425 FOR X1=X TO 99
03430 IF P$(X1+1)<>" "THEN P$(X1)=P$(X1+1)
03435 IF LEFT$(P$(X1),3)=" 99"THEN RETURN
03440 NEXT X1
03445 RETURN
03500 REM ROUTINE TO PUT DATA BASE ON TAPE FILE
03505 LINE INPUT "ENTER A UNIQUE TITLE FOR THIS FILE: ";Z$
03510 PRINT "PLACE A BLANK TAPE IN YOUR RECORDER & PRESS 'RECORD'
."
03515 PRINT "HIT 'RETURN' WHEN READY."
03520 CNTRL 4,1:PAUSE
03525 GOSUB 65110
03530 STOP
03535 PRINT :CNTRL 4,0:RETURN
03600 REM ROUTINE TO GET NEW DATA FILE
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CALENDER - WILKINSON < CONT'D >

```
03605 PRINT "NOTE THAT IF YOU CONTINUE, YOUR CURRENT LIST WILL BE
      LOST."
03610 LINE INPUT "DO YOU WISH TO CONTINUE? ";A$
03615 IF LEFT$(A$,1)<>"Y"THEN RETURN
03620 PRINT "ENTER THE TITLE OF THE FILE DESIRED (HITTING 'RETURN
      ' WILL"
03625 LINE INPUT "LOAD THE FIRST FILE ENCOUNTERED): ";Z$
03630 PRINT "PLACE THE CASSETTE IN YOUR PLAYER & PRESS 'PLAY'."
03635 PRINT "HIT 'RETURN' WHEN READY."
03640 CNTRL 4,1:PAUSE
03645 GOSUB 65010
03650 STOP
03655 PRINT :CNTRL 4,0:GOSUB 3010:GOTO 2010
04010 F2=1:GOSUB 50:PRINT :REM DRAW CALENDAR
04015 IF F=1AND D>M(N)THEN RETURN
04020 PRINT :FOR X1=0TO 99
04030 IF VAL(LEFT$(P$(X1),3))=M AND Y=VAL(MID$(P$(X1),10,4))THEN
PRINT P$(X1)
04035 IF LEFT$(P$(X1),3)=" 99"THEN RETURN :REM SPEED THINGS UP
04040 NEXT X1
04050 RETURN
04055 REM FROM LINE 437
04060 D$=STR$(D):S=5
04070 IF D<10THEN D$=" "+STR$(D)
04075 IF D=10THEN S=6
04080 FOR X2=0TO 99
04090 IF Y<>VAL(MID$(P$(X2),10,4))THEN 4120
04100 IF M<>VAL(LEFT$(P$(X2),3))THEN 4120
04101 IF F5=1THEN 4140
04102 IF F4<>1THEN 4110
04103 IF Z7=0AND D$=MID$(P$(X2),5,4)THEN F4=0:PRINT MID$(D$,2,2)+
      "*"::GOTO 445
04104 IF VAL(MID$(P$(X2),5,4))=D THEN PRINT SPC((Z7*10)-2)LEFT$(D
      $,3)+"*":F4=0:GOTO 445
04105 IF D=1THEN 4120
04110 IF D$=MID$(P$(X2),5,4)THEN PRINT SPC(S)LEFT$(D$,3)+"*":GOT
      O 445
04120 NEXT X2:IF F4=1THEN 4160
04125 IF F5=1THEN F6=0:RETURN
04130 PRINT SPC(S)D$::GOTO 445
04140 IF D=VAL(MID$(P$(X2),5,4))THEN F6=1:RETURN
```





<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RESONANT FREQ/CALC - BORDEN

```
00010 GOSUB 630
00020 PRINT "RESONANT FREQUENCY CALCULATOR      VER 1      7/10/78"
00030 REM      BY KAREN BORDEN
00040 REM      5201 ASH
00050 REM      ROELAND PARK, KANSAS 66205.
00060 GOSUB 640
00070 PRINT " FOR CYCLES, YOU MAY ENTER, : CYCLES, CY, HERTZ, OR
HZ."
00080 PRINT "FOR KILOCYCLES, YOU MAY USE: KC, KHZ, KILOCYCLES, OR
KILOHERTZ"
00090 PRINT "THE SAME FOR OTHER UNITS.  THE PROGRAM READS ONLY TH
E FIRST TWO"
00100 PRINT "LETTERS.  :EXAMPLE: 2600 CYCLES, ENTER 2600 ,PRESS R
ETURN. THEN CY."
00110 PRINT :PRINT
00120 PRINT "SPECIFY WHICH FACTOR YOU ARE SOLVING FOR, "
00130 PRINT "    FREQUENCY= F      CAPACITY = C      INDUCTANCE = L"
00140 CLEAR
00150 LINE INPUT :A$
00160 GOSUB 640
00170 P4=39.47841
00180 P2=6.283185
00190 IF A$="F" THEN GOTO 230
00200 IF A$="C" THEN GOTO 400
00210 IF A$="L" THEN GOTO 530
00220 PRINT "ERROR,.....ENTER A SINGLE LETTER AS FUNCTION !
":GOTO 120
00230 REM
00240 PRINT "SOLVING FOR FREQUENCY !":PRINT
00250 GOSUB 650
00260 GOSUB 750
00270 R2=SQR(VAL(Y1$)*VAL(Y2$))
00280 C2=VAL(C2$):L2=VAL(L2$)
00290 REM
00300 F2=P2*SQR(C2*L2)
00310 F3=R2/F2
00320 GOSUB 630
00330 PRINT "FOR VALUES:      ";C2$;" ";U2$;"  AND ";L2$;" ";U3$
00340 PRINT "      RESONANT FREQUENCY IS ";
00350 IF F3<=1000 THEN PRINT F3;" CYCLES (HZ.)"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RESONANT FREQ/CALC - BORDEN < CONT'D >

```
00360 IF F3>1000AND F3<=1.0E+07THEN F3=F3/1000:PRINT F3;" KILOCYC  
LES (KHZ.)"  
00370 IF F3>1.00E+07THEN F3=F3/1000000:PRINT F3;" MEGACYCLES (M  
HZ.)"  
00380 GOSUB 640  
00390 GOTO 120  
00400 REM  
00410 PRINT "SOLVING FOR CAPACITY":PRINT  
00420 GOSUB 830  
00430 GOSUB 750  
00440 R2=1/(VAL(Y3$)*VAL(Y3$))*VAL(Y2$)  
00450 C2=P4*(VAL(F2$)^2)*VAL(L2$)  
00460 C3=R2/C2  
00470 PRINT  
00480 IF C3>=1THEN PRINT , "CAPACITOR VAL. IS ";C3;" FARADS."  
00490 IF C3<1AND C3>=1.E-10THEN C3=C3*1.E+6:PRINT , "CAPACITOR VAL  
. IS ";C3;" MFD"  
00500 IF C3<1.E-10THEN C3=C3*1.E+12:PRINT , "CAPACITOR VALUE IS. "  
;C3;" PF."  
00510 GOSUB 640  
00520 GOTO 120  
00530 PRINT "SOLVING FOR INDUCTANCE":PRINT  
00540 GOSUB 830  
00550 GOSUB 650  
00560 R2=1/(VAL(Y3$)*VAL(Y3$))*VAL(Y1$)  
00570 L2=P4*(VAL(F2$)^2)*VAL(C2$)  
00580 L3=R2/L2  
00590 IF L3>=1THEN PRINT , "INDUCTOR VAL. IS ";L3;" HENRYS."  
00600 IF L3<1AND L3>=.001THEN L3=L3*1000:PRINT , "INDUCTOR VAL. IS  
";L3;" MH."  
00610 IF L3<.001THEN L3=L3*1.0E+6:PRINT , "INDUCTOR VALUE. IS ";L3  
;" UH."  
00620 PRINT :PRINT :GOTO 120  
00630 FOR I=1TO 10:PRINT :NEXT I:RETURN  
00640 FOR I=1TO 5:PRINT :NEXT I:RETURN  
00650 LINE INPUT " ENTER CAPACITOR VALUE ? ";C2$  
00660 IF VAL(C2$)=0THEN PRINT , " ERROR, RE-ENTER,.....  
":GOTO 650  
00670 LINE INPUT " ENTER> PF, MMF, MFD, FARADS, ? ";U2$  
00680 IF LEFT$(U2$,2)="PF"THEN Y1$="1.00E+12":RETURN  
00690 IF LEFT$(U2$,2)="MF"THEN Y1$="1.00E+06":RETURN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RESONANT FREQ/CALC - BORDEN &lt; CONT'D &gt;

```

000700 IF LEFT$(U2$,2)="FA"THEN V1$="1.00":RETURN
000710 IF LEFT$(U2$,2)="FD"THEN V1$="1.00":RETURN
000720 IF LEFT$(U2$,2)="MM"THEN V1$="1.00E+12":RETURN
000730 IF LEFT$(U2$,2)="UU"THEN V1$="1.00E+12":RETURN
000740 PRINT ,U2$;" UNFAMILIAR TERM, RE-ENTER....":GOTO 650
000750 LINE INPUT " ENTER INDUCTOR VALUE ? ";L2$
000760 IF VAL(L2$)=0THEN PRINT ,"ERROR, RE-ENTER....
":GOTO 750
000770 LINE INPUT "ENTER> UH, MH, HENRYS, ? ";U3$
000780 IF LEFT$(U3$,2)="UH"THEN V2$="1.00E+06":RETURN
000790 IF LEFT$(U3$,2)="MH"THEN V2$="1.00E+03":RETURN
000800 IF LEFT$(U3$,2)="HE"THEN V2$="1.00":RETURN
000810 IF LEFT$(U3$,2)="HV"THEN V2$="1.00":RETURN
000820 PRINT ,U3$;" UNFAMILIAR TERM, RE-ENTER.....":GOTO 750
000830 LINE INPUT " ENTER DESIRED FREQUENCY ? ";F2$
000840 IF VAL(F2$)=0THEN PRINT ,"ERROR, RE-ENTER":PRINT :PR
INT :GOTO 830
000850 LINE INPUT " ENTER> CYCLES= CY KILOCYCLES= KC MEGACYCLE
S= MC ? ";U4$
000860 IF LEFT$(U4$,2)="CY"THEN V3$="1.0":RETURN
000870 IF LEFT$(U4$,2)="HZ"THEN V3$="1.0":RETURN
000880 IF LEFT$(U4$,2)="KC"THEN V3$="1000":RETURN
000890 IF LEFT$(U4$,2)="KH"THEN V3$="1000":RETURN
000900 IF LEFT$(U4$,2)="KI"THEN V3$="1000":RETURN
000910 IF LEFT$(U4$,2)="ME"THEN V3$="1.0E+06":RETURN
000920 IF LEFT$(U4$,2)="MH"THEN V3$="1.0E+06":RETURN
000930 IF LEFT$(U4$,2)="MC"THEN V3$="1.0E+06":RETURN
000940 PRINT ,U4$;" UNFAMILIAR TERM, RE-ENTER....":GOTO 830
000950 ,76
000960 FOR I=17000TO 24000:X1=PEEK(I):X2=PEEK(I+1):X3=PEEK(I+2)
000970 IF X1=0AND X2=19AND X3=0THEN PRINT I:STOP
000980 NEXT I

```

14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: PAD CALCULATOR - BORDEN

```
00010 PRINT :PRINT :PRINT
00020 PRINT , "PAD 1      ATTENUATION PAD IMPEDIENCE CALCULATION"
00030 PRINT TAB(30) "BY      KAREN BORDEN,"
00040 PRINT TAB(30) "      5201 ASH"
00050 PRINT TAB(30) "      ROELAND PARK, KANSAS.      66205"
00060 PRINT
00070 PRINT , "ENTER VALUES FOR R1, AND LOAD (R4),      R2 WILL BE CA
LCULATED"
00080 PRINT , "R1 AND R3 MUST BE EQUIL !"
00090 PRINT :PRINT
00100 LINE INPUT " DRAW DIAGRAM ?  Y/N  ";A$
00110 PRINT :PRINT
00120 IF A$="N" THEN 260
00130 PRINT "SIG. IN  ->      (*)- - - - -*- - - -[ R2 ]- - - -*- -
- -(*)  <--*"
00140 GOSUB 220
00150 GOSUB 220
00160 PRINT TAB(23) "[ R1 ]";TAB(43) "[ R3 ]";TAB(60) "[ R4 ]  LOAD"
00170 GOSUB 220
00180 GOSUB 220
00190 PRINT "  COMMON ->      (*)- - - - -*- - - - - - - - - - - - - - - *
- -(*)  <--*"
00200 PRINT :PRINT
00210 GOTO 240
00220 PRINT TAB(25) "!" ;TAB(46) "!" ;TAB(62) "!"
00230 RETURN
00240 PRINT "ENTER VALUE:"
00250 PRINT
00260 INPUT "R1= ";R1
00270 PRINT "R2= UNKNOWN"
00280 PRINT "R3=" ;R1
00290 R3=R1
00300 INPUT "LOAD, R4= ";R4
00310 IF R4>=R1 THEN 410
00320 IF R2=0 THEN GOTO 330
00330 L1=(1/R1)+(1/R4)
00340 L2=1/L1
00350 L3=(1/R4)-(1/R1)
00360 L4=1/L3
00370 L5=L4-L2
00380 R2=L5
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: PAD CALCULATOR - BORDEN < CONT'D >

```

00390 IF R2>0 THEN PRINT "      R2 SHOULD BE....";R2;"OHMS !"
00400 PRINT
00410 IF R4>=R1 THEN R2=0
00420 IF R2<=0 THEN PRINT "NOT POSSIBLE WITH THE VALUES GIVEN"
      "
00430 S1=1/R4+1/R3
00440 S2=1/S1
00450 S3=S2+R2
00460 E3=100*(R2/S3)
00470 E2=100-E3
00480 IF R2>0 THEN PRINT "      % OF SIGNAL OUTPUT IS ";E2;"%"
00490 E4=100/E2
00500 D1=20*(LOG(E4)/LOG(10))
00510 PRINT "      SIGNAL RATIO IS ";E4;" TO 1"
00520 PRINT "      ATTENUATION IS ";D1;" D.B."
00530 PRINT :PRINT :PRINT
00540 GOTO 260

```

12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: WEATHER RECORDS - MYRUP

```
00010 PRINT "                WEATHER RECORDS PROGRAM"
00020 DIM S5(12,4)
00030 GOTO 1870
00040 PRINT CHR$(7)
00050 LINE INPUT "DO YOU WANT TO INPUT FROM A TAPE FILE(Y/N)? ";K
00060 IF K$="Y" THEN GOTO 2040
00070 PRINT CHR$(7)
00080 PRINT "IN THIS PROGRAM, I RECOMMEND THE USE OF DEGREES FAHR
ENHEIT FOR"
00090 PRINT "TEMPERATURE, MILLIBARS FOR SEA-LEVEL PRESSURE AND INC
HES FOR PRECIPITATION"
00100 PRINT CHR$(7):LINE INPUT "WHAT MONTH? ";M$
00110 PRINT CHR$(7):INPUT "WHAT YEAR? ";Y1
00120 IF M$="JANUARY" THEN LET I=1
00130 IF M$="FEBRUARY" THEN LET I=2
00140 IF M$="MARCH" THEN LET I=3
00150 IF M$="APRIL" THEN LET I=4
00160 IF M$="MAY" THEN LET I=5
00170 IF M$="JUNE" THEN LET I=6
00180 IF M$="JULY" THEN LET I=7
00190 IF M$="AUGUST" THEN LET I=8
00200 IF M$="SEPTEMBER" THEN LET I=9
00210 IF M$="OCTOBER" THEN LET I=10
00220 IF M$="NOVEMBER" THEN LET I=11
00230 IF M$="DECEMBER" THEN LET I=12
00240 PRINT CHR$(7):IF M$="TEST" THEN INPUT "HOW MANY TEST POINTS?
";D
00250 IF I>0 THEN LET D=S5(I,1)
00260 DIM T1(D),T2(D),P(D),F(D),U(D)
00270 LET D1=D:LET D2=D:LET D3=D
00280 IF D=0 THEN GOTO 1550
00290 PRINT CHR$(7)
00300 PRINT "PLEASE INPUT MAXIMUM TEMPERATURE FOR EACH DAY OF THE
MONTH"
00310 PRINT "INDICATE MISSING DATA BY ENTERING '00'"
00320 FOR X=1 TO D
00330 PRINT X,
00340 INPUT T1(X)
00350 IF T1(X)=00 THEN LET D1=D1-1
00360 NEXT X
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: WEATHER RECORDS - MYRUP < CONT'D >

```
00370 PRINT CHR$(7)
00380 PRINT "PLEASE INPUT MINIMUM TEMPERATURE FOR EACH DAY OF THE
MONTH"
00390 FOR X=1TO D
00400 PRINT X,
00410 INPUT T2(X)
00420 IF T2(X)=00THEN LET D2=D2-1
00430 NEXT X
00440 PRINT CHR$(7)
00450 PRINT "PLEASE INPUT PRESSURE FOR EACH DAY OF THE MONTH"
00460 FOR X=1TO D
00470 PRINT X,
00480 INPUT P(X)
00490 IF P(X)=0THEN LET D3=D3-1
00500 NEXT X
00510 PRINT CHR$(7)
00520 PRINT "PLEASE INPUT PRECIPITATION FOR EACH DAY (ZERO IF NON
E)"
00530 FOR X=1TO D
00540 PRINT X,
00550 INPUT F(X)
00560 NEXT X
00570 FOR X=1TO D
00580 LET U(X)=INT(10*F(X))
00590 NEXT X
00600 PRINT CHR$(7)
00610 PRINT TAB(10)"INPUT DATA FOR ";M$," ";Y1
00620 PRINT "DAY","MAX TEMP","MIN TEMP","PRESSURE","PRECIP"
00630 FOR X=1TO D
00640 PRINT X,T1(X),T2(X),P(X),F(X):NEXT X
00650 PRINT CHR$(7)
00660 LINE INPUT "IS ALL THE DATA OK(Y/N)?":R$
00670 IF R$="N"THEN GOTO 1290
00680 PRINT CHR$(7)
00690 PRINT "          PLOT OF INPUT DATA"
00700 GOTO 1580
00710 PRINT CHR$(7):PRINT CHR$(7)
00720 PRINT "NOW COMPUTING WEATHER STATISTICS FOR ";M$," ";Y1
00730 FOR X=1TO D
00740 LET B1=B1+T1(X):LET A1=B1/D1
00750 LET B2=B2+T2(X):LET A2=B2/D2
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: WEATHER RECORDS - MYRUP < CONT'D >

```
00760 LET B3=B3+P(X):LET A3=B3/D3
00770 LET B4=B4+F(X)
00780 NEXT X
00790 FOR X=1 TO D
00800 IF T1(X)=0 THEN GOTO 860
00810 LET E1=E1+(ABS(T1(X)-A1))2
00820 IF T2(X)=0 THEN GOTO 860
00830 LET E2=E2+(ABS(T2(X)-A2))2
00840 IF P(X)=0 THEN GOTO 860
00850 LET E3=E3+(ABS(P(X)-A3))2
00860 NEXT X
00870 LET S1=(E1/(D1-1)).5
00880 LET S2=(E2/(D2-1)).5
00890 LET S3=(E3/(D3-1)).5
00900 FOR X=1 TO D
00910 IF T1(X)=0 THEN GOTO 960
00920 LET H1=H1+X*T1(X)
00930 LET J1=J1+X*X
00940 LET K1=K1+X: LET L1=K1/D1
00950 LET M1=(H1-D1*A1*L1)/(J1-D1*L1*L1)
00960 NEXT X
00970 LET N1=M1*D
00980 FOR X=1 TO D
00990 IF T2(X)=0 THEN GOTO 1040
01000 LET H2=H2+X*T2(X)
01010 LET J2=J2+X*X
01020 LET K2=K2+X: LET L2=K2/D2
01030 LET M2=(H2-D2*A2*L2)/(J2-D2*L2*L2)
01040 NEXT X
01050 LET N2=M2*D
01060 FOR X=1 TO D
01070 IF P(X)=0 THEN GOTO 1120
01080 LET H3=H3+X*P(X)
01090 LET J3=J3+X*X
01100 LET K3=K3+X: LET L3=K3/D3
01110 LET M3=(H3-D3*A3*L3)/(J3-D3*L3*L3)
01120 NEXT X
01130 N3=N3*D:PRINT CHR$(7):PRINT CHR$(7):PRINT CHR$(7)
01140 PRINT "          STATISTICS FOR ";M$;"", "Y1
01150 PRINT "NUMBERS IN PARENTHESES ARE THE CLIMATOLOGICAL NORMAL
S FOR DAVIS, CALIFORNIA"
01160 PRINT "MAX TEMP", "MIN TEMP", "PRESSURE"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: WEATHER RECORDS - MYRUP < CONT'D >

```
01170 PRINT "MEAN",A1;"(";"S5(I,2);")";
01180 PRINT TAB(41)A2;"(";"S5(I,3);")";
01190 PRINT TAB(69)A3
01200 PRINT "STANDARD"
01210 PRINT "DEVIATION",S1,,S2,,S3
01220 PRINT "TREND",N1,,N2,,N3
01230 PRINT "DATA "
01240 PRINT "POINTS",D1,,D2,,D3
01250 PRINT
01260 PRINT "TOTAL PRECIPITATION FOR ";M$;" : ";B4;"(";"S5(I,4);")
"
01270 IF K8$="Y"THEN GOTO 2090
01280 STOP
01290 PRINT CHR$(7):INPUT "WHAT DAY IS THE ERROR? ";R1
01300 LINE INPUT "ERROR IN MAX TEMP(1),MIN TEMP(2),PRESSURE(3) OR
PRECIP(4) ";E$
01310 IF E$="1"THEN GOTO 1350
01320 IF E$="2"THEN GOTO 1400
01330 IF E$="3"THEN GOTO 1450
01340 IF E$="4"THEN GOTO 1500
01350 IF T1(R1)=0THEN LET D1=D1+1
01360 INPUT "ENTER CORRECT TEMPERATURE";T1(R1):IF T1(R1)=0THEN D1
=D1-1
01370 PRINT CHR$(7):LINE INPUT "ANY OTHER ERRORS(Y/N)? ";T$
01380 IF T$="Y"THEN GOTO 1290
01390 IF T$="N"THEN GOTO 610
01400 IF T2(R1)=0THEN LET D2=D2+1
01410 INPUT "ENTER CORRECT TEMPERATURE ";T2(R1):IF T2(R1)=0THEN
D2=D2-1
01420 LINE INPUT "ANY OTHER ERRORS(Y/N)? ";Y$
01430 IF Y$="Y"THEN GOTO 1290
01440 IF Y$="N"THEN GOTO 610
01450 IF P(R1)=0THEN LET D3=D3+1
01460 INPUT "ENTER CORRECT PRESSURE ";P(R1):IF P(R1)=0THEN D3=D3
-1
01470 LINE INPUT "ANY OTHER ERRORS(Y/N)? ";P$
01480 IF P$="Y"THEN GOTO 1290
01490 IF P$="N"THEN GOTO 610
01500 INPUT "ENTER CORRECT PRECIP";F(R1)
01510 LINE INPUT "ANY OTHER ERRORS(Y/N)? ";Q$
01520 IF Q$="Y"THEN GOTO 1290
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: WEATHER RECORDS - MYRUP < CONT'D >

```
01530 IF Q$="N"THEN GOTO 570
01540 STOP
01550 PRINT "HELP!!! I DON'T KNOW WHAT TO DO! PLEASE START AGAIN."
01560 PRINT "(AND WATCH YOUR SPELLING!)"
01570 STOP
01580 FOR X=1TO D
01590 PRINT X;
01600 IF T2(X)=0GOTO 1640
01610 IF T2(X)<25THEN PRINT "C";
01620 IF T2(X)<25THEN GOTO 1640
01630 PRINT TAB((T2(X)-25)/2)"C";
01640 IF T1(X)=0GOTO 1660
01650 PRINT TAB((T1(X)-25)/2)"H";
01660 PRINT TAB(30);
01670 FOR Y=1TO U(X)
01680 IF U(X)=0GOTO 1700
01690 PRINT "*";
01700 NEXT Y
01710 IF P(X)=0GOTO 1730
01720 PRINT TAB(P(X)-955)"P":GOTO 1740
01730 PRINT
01740 NEXT X
01750 PRINT TAB(2)"30";TAB(7)"40";TAB(12)"50";TAB(17)"60";TAB(22)
"70"
01760 PRINT TAB(22)"70";TAB(27)"80";TAB(32)"90";TAB(37)"100"
01770 PRINT "PRECIP: *=0.1";
01780 PRINT TAB(45)"1000";TAB(55)"1010";TAB(65)"1020";TAB(75)"103
0"
01790 PRINT CHR$(7):LINE INPUT "REPEAT DATA PLOT(Y/N)? ";0$
01800 IF 0$="Y"GOTO 1580
01810 PRINT CHR$(7):LINE INPUT "ANY MORE CORRECTIONS(Y/N)? ";01$
01820 IF 01$="Y"THEN GOTO 1290
01830 PRINT CHR$(7)
01840 LINE INPUT "DO YOU WANT TO WRITE A DATA TAPE FILE NOW(Y/N)?
";09$
01850 IF 09$="Y"THEN GOTO 2090
01860 GOTO 710
01870 FOR X=1TO 12
01880 READ S5(X,1)
01890 NEXT X
```





<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: LOWPASS DESIGN - DACEY

```
00010 REM >>>>>> OPERATIONAL AMPLIFIER FILTER DESIGN PROGRAM <<
<<<<<
00020 REM                LOWPASS FILTER OR HIGHPASS FILTER
00030 REM                COPYRIGHT 1978 BY FRANK DACEY
00040 REM                //////////////////////////////////////
////
00050 REM *****      BY :      FRANK DACEY                        **
*****
00060 REM *****                328 BARBARA DRIVE                    **
*****
00070 REM *****                POINT PLEASANT, NJ 08742            **
*****
00080 REM                //////////////////////////////////////
////
00090 P1=3.14159:M=1000000:N=1000:L1$=" MICROFARADS":L2$=" PICOFA
RADS"
00100 L3$=" K OHMS":FOR I=1TO 6:PRINT :NEXT I:PRINT TAB(5)"*****";
TAB(15);
00110 PRINT "OPERATIONAL AMPLIFIER FILTER DESIGN PROGRAM          ***
*"
00120 PRINT TAB(20)"LOWPASS FILTER OR HIGHPASS FILTER"
00130 PRINT :PRINT "DURING THE PROGRAM THERE WILL BE OCCASIONAL P
AUSES"
00140 PRINT "TO ALLOW TIME FOR READING AND/OR DRAWING SCHEMATICS.
TO"
00150 PRINT "RESTART THE PROGRAM --- PRESS THE SPACE BAR."
00160 PRINT :PRINT "DO YOU WANT INSTRUCTIONS OR DO YOU WANT TO EN
TER"
00170 LINE INPUT "THE PROGRAM ? (I=INSTRUCTIONS/P=PROGRAM)      ";Q$
00180 IF LEFT$(Q$,1)="P" GOTO 430
00190 PRINT :PRINT
00200 PRINT "THIS PROGRAM, WITH YOUR ASSISTANCE, WILL DESIGN LOWP
ASS"
00210 PRINT "AND/OR HIGHPASS FILTER CIRCUITS USING OPERATIONAL"
00220 PRINT "AMPLIFIERS. THESE FILTERS ARE UNITY GAIN TYPES WITH
ABOUT"
00230 PRINT "A NINE DB/OCTAVE ROLLOFF."
00240 PRINT
00250 PRINT "YOU WILL BE ASKED THREE THINGS:"
00260 PRINT TAB(5)"1. TYPE OF FILTER DESIRED - LOWPASS OR HIGHPAS
S."
00270 PRINT TAB(5)"2. THE DESIRED CUTOFF FREQUENCY.(3 DB DOWN FRE
QUENCY)"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: LOWPASS DESIGN - DACEY

```
00280 PRINT TAB(5)"3. THE MINIMUM INPUT IMPEDANCE REQUIRED."
00290 PRINT
00300 PAUSE
00310 PRINT TAB(5)"***REMEMBER - YOU MUST ALSO CONNECT PLUS AND M
INUS***"
00320 PRINT TAB(5)"***POWER SUPPLY VOLTAGES IN ADDITION TO THE OT
HER ***"
00330 PRINT TAB(5)"***COMPONENTS SHOWN IN THE SCHEMATIC. ANY VOL
TAGE***"
00340 PRINT TAB(5)"***BETWEEN ABOUT EIGHT TO FIFTEEN VOLTS WILL W
ORK.***"
00350 PRINT :PRINT TAB(5)"*** PINOUTS FOR THE 741 OP-AMP ***"
00360 PRINT TAB(5)"** PIN 4 MINUS INPUT * PIN 5 PLUS INPUT * PIN
6"
00370 PRINT TAB(5)"** MINUS SUPPLY VOLTAGE * PIN 10 OUTPUT * PIN
11"
00380 PRINT TAB(5)"** POSITIVE SUPPLY VOLTAGE. (FOR 14 PIN DIP)"
00390 PRINT
00400 PRINT TAB(5)"***YOUR CHOICE OF CUTOFF FREQUENCIES SHOULD BE
***"
00410 PRINT TAB(5)"***NO GREATER THAN 500 KHZ FOR THIS PROGRAM.
***";
00420 PAUSE
00430 PRINT :PRINT :PRINT
00440 PRINT "ARE YOU INTERESTED IN DESIGNING A LOWPASS FILTER OR"
00450 LINE INPUT "A HIGHPASS FILTER ? (L=LOWPASS//H=HIGHPASS) ";Q
$
00460 PRINT :PRINT "HERE IS THE SCHEMATIC OF THE CIRCUIT YOU CHOS
E."
00470 PRINT "COPY THE CIRCUIT AND CONNECT ALL THE DOTTED LINES."
00480 IF LEFT$(Q$,1)="L" GOTO 720
00490 IF LEFT$(Q$,1)="H" GOTO 930
00500 GOTO 440
00510 GOSUB 1170
00520 R1=R:R2=R
00530 PRINT "THE VALUE OF R1 IS ";R1;L3$
00540 PRINT "THE VALUE OF R2 IS ";R2;L3$
00550 C1=INT(M/(P1*F2*R)):C3=C1
00560 GOSUB 1270
00570 PRINT "THE VALUE OF C1 IS ";C3$
00580 C2=INT(M/(4*P1*F1*R)):C3=C2
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: LOWPASS DESIGN - DACEY < CONT'D >

```
00590 GOSUB 1270
00600 PRINT "THE VALUE OF C2 IS ";C3$
00610 GOTO 1480
00620 GOSUB 1170
00630 R1=INT((1.5*R*10)+.5)/10
00640 C1=INT(M/(4*P1*F1*R1)):C3=C1
00650 R2=2*R1
00660 GOSUB 1270
00670 PRINT "THE VALUE OF R1 IS ";R1;L3$
00680 PRINT "THE VALUE OF R2 IS ";R2;L3$
00690 PRINT "THE VALUE OF C1 IS ";C3$
00700 PRINT "THE VALUE OF C2 IS ";C3$
00710 GOTO 1480
00720 PRINT :PRINT :PRINT TAB(5);
00730 PRINT "***** OPERATIONAL AMPLIFIER UNITY GAIN LOWPASS FILTER
*****"
00740 PRINT :PRINT :F$="LOWPASS"
00750 PRINT TAB(29)"!";:FOR I=1TO 30:PRINT "-";:NEXT I:PRINT "!"
00760 PRINT TAB(20)"! ! C1";TAB(29)"!";TAB(33)"!-";TAB(60)"!"
00770 PRINT TAB(3)"X";:GOSUB 920:PRINT "[ R1 ]";:GOSUB 920:PRINT
"!";
00780 GOSUB 920:PRINT "! !";:GOSUB 920:GOSUB 920:PRINT "!";
00790 GOSUB 920:PRINT "!";TAB(35)"-";TAB(41)"-";TAB(60)"!"
00800 PRINT TAB(16)"!";TAB(20)"! !";TAB(33)"!";TAB(48)"-";TAB(60)
"!";
00810 PRINT TAB(2)"IN";TAB(16)"!";TAB(33)"!";TAB(36)"TYPE 741 OP-
AMP";
00820 PRINT TAB(53)"-";:GOSUB 920:GOSUB 920:PRINT "!---X"
00830 PRINT TAB(16)"!";TAB(33)"!";TAB(48)"-";TAB(63)"OUT"
00840 PRINT TAB(3)"X";TAB(16)"!";:GOSUB 920:PRINT "[ R2 ]";:GOSUB
920
00850 PRINT "!";:GOSUB 920:PRINT "!";TAB(35)"+";TAB(41)"-";TAB(64)
)"X"
00860 PRINT TAB(3)"!";TAB(29)"!";TAB(33)"!-";TAB(64)"!";PAUSE
00870 PRINT TAB(3)"!";TAB(28);:GOSUB 920:PRINT TAB(64)"!"
00880 PRINT TAB(3)"!";TAB(28);:GOSUB 920:PRINT " C2";TAB(64)"!"
00890 PRINT TAB(3)"!";TAB(29)"!";TAB(64)"!"
00900 PRINT TAB(2)"GND";TAB(28)"GND";TAB(63)"GND"
00910 PRINT :PRINT :FOR I=1TO 70:PRINT "*";:NEXT I:GOTO 510
00920 PRINT "---";:RETURN
00930 PRINT :PRINT :PRINT TAB(5);
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: LOWPASS DESIGN - DACEY < CONT'D >

```
00940 PRINT "**** OPERATIONAL AMPLIFIER UNITY GAIN HIGHPASS FILTE
R ****"
00950 PRINT :PRINT :F$="HIGHPASS"
00960 PRINT TAB(29)"!";:FOR I=1TO 30:PRINT "-";:NEXT I:PRINT "!"
00970 PRINT TAB(7)"! ! C1";TAB(29)"!";TAB(33)"!-";TAB(60)"!"
00980 PRINT TAB(3)"X";:GOSUB 1160:PRINT "! !";:GOSUB 1160:GOSUB 1
160
00990 PRINT "!";:GOSUB 1160:PRINT "[ R1 ]";:GOSUB 1160:PRINT "!";
01000 GOSUB 1160:PRINT "!";TAB(35)"-";TAB(41)"-";TAB(60)"!"
01010 PRINT TAB(7)"! !";TAB(16)"!";TAB(33)"!";TAB(48)"-";TAB(60)"
!"
01020 PRINT TAB(2)"IN";TAB(16)"!";TAB(33)"!";TAB(36)"TYPE 741 OP-
AMP";
01030 PRINT TAB(53)"-";:GOSUB 1160:GOSUB 1160:PRINT "!---X"
01040 PRINT TAB(16)"!";TAB(20)"! ! C2";TAB(33)"!";TAB(48)"-";TAB(
63)"OUT"
01050 PRINT TAB(3)"X";TAB(16)"!";:GOSUB 1160:PRINT "! !";:GOSUB 1
160
01060 GOSUB 1160:PRINT "!";:GOSUB 1160:PRINT "!";TAB(35)"+";
01070 PRINT TAB(41)"-";TAB(64)"X"
01080 PRINT TAB(3)"!";TAB(20)"! !";TAB(29)"!";TAB(33)"!-";TAB(64)
"!"
01090 PAUSE :PRINT TAB(3)"!";TAB(28);:GOSUB 1160:PRINT TAB(64)"!"
01100 PRINT TAB(3)"!";TAB(28)"!R!";TAB(64)"!"
01110 PRINT TAB(3)"!";TAB(28)"!2!";TAB(64)"!"
01120 PRINT TAB(3)"!";TAB(28);:GOSUB 1160:PRINT TAB(64)"!"
01130 PRINT TAB(3)"!";TAB(29)"!";TAB(64)"!"
01140 PRINT TAB(2)"GND";TAB(28)"GND";TAB(63)"GND"
01150 PRINT :PRINT :FOR I=1TO 70:PRINT "*";:NEXT I:GOTO 620
01160 PRINT "---";:RETURN
01170 PRINT :PRINT :F=0:R=0
01180 INPUT "SPECIFY THE DESIRED CUTOFF FREQUENCY IN KILOHERTZ.
";F
01190 INPUT "SPECIFY THE MINIMUM INPUT IMPEDANCE IN K-OHMS.
";R
01200 F1=F*.707
01210 F2=F*1.414
01220 PRINT :PRINT
01230 PRINT "HERE ARE THE COMPONENT VALUES FOR THE ";F$;" FILTER
WHEN"
01240 PRINT "THE CUTOFF FREQUENCY IS ";F;" KHZ."
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: LOWPASS DESIGN - DACEY &lt; CONT'D &gt;

```

01250 PRINT
01260 RETURN
01270 C3$="":C4$="":C5$=""
01280 IF C3<1000 THEN C3$=STR$(C3):C3$=C3$+L2$:RETURN
01290 IF C3>=1000 THEN C3=INT(C3/N)
01300 C4$=STR$(C3)
01310 X=LEN(C4$):X1=X-1
01320 IF C3<10 THEN C3$=" 0.00":GOTO 1360
01330 IF C3<100 THEN C3$=" 0.0" :GOTO 1360
01340 IF C3<1000 THEN C3$=" 0." :GOTO 1360
01350 PRINT TAB(20)"*****END OF THE PROGRAM*****"
01360 FOR I=2 TO X1
01370 C5$=MID$(C4$,I,1)
01380 C3$=C3$+C5$
01390 NEXT I
01400 C3$=C3$+L1$
01410 RETURN
01420 C3$="":FOR I=1 TO 2
01430 C5$=MID$(C4$,I,1)
01440 C3$=C3$+C5$
01450 NEXT I
01460 C3$=C3$+L1$
01470 RETURN
01480 PRINT TAB(5)"DO YOU WANT TO:"
01490 PRINT TAB(13)"1. DESIGN ANOTHER FILTER CIRCUIT ?"
01500 PRINT TAB(13)"2. END THE PROGRAM ?"
01510 PRINT
01520 PRINT TAB(5);
01530 LINE INPUT "TYPE D FOR DESIGN OR E FOR END ";Q$
01540 IF LEFT$(Q$,1)="D" GOTO 430
01550 FOR I=1 TO 5:PRINT :NEXT I:PRINT TAB(20)"**END OF PROGRAM**"
01560 PRINT :PRINT :PRINT
01570 END

```

19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: BANDPASS DESIGN - DACEY

```
00010 REM      >>>>>> OPERATIONAL AMPLIFIER FILTER DESIGN PROGRAM <
<<<<<<
00020 REM      >>>>>>          BANDPASS FILTER OR NOTCH FILTER      <
<<<<<<
00030 REM      >>>>>>          COPYRIGHT 1978 BY FRANK DACEY        <
<<<<<<
00040 REM      //////////////////////////////////////
////////
00050 REM      *****      BY :      FRANK DACEY                  *
*****
00060 REM      *****          328 BARBARA DRIVE                  *
*****
00070 REM      *****          POINT PLEASANT, NJ 08742          *
*****
00080 REM      //////////////////////////////////////
////////
00090 P1=3.14159:M=1000000:N=10000:L1$=" MICROFARADS":L2$=" PICOF
ARADS"
00100 FOR I=1 TO 6:PRINT :NEXT I:PRINT TAB(5)"*****";TAB(15);
00110 PRINT "OPERATIONAL AMPLIFIER FILTER DESIGN PROGRAM      ***
**"
00120 PRINT TAB(24)"BANDPASS OR NOTCH FILTER"
00130 PRINT :PRINT "DURING THE PROGRAM THERE WILL BE OCCASIONAL P
AUSES"
00140 PRINT "TO ALLOW TIME FOR READING AND/OR DRAWING SCHEMATICS.
TO "
00150 PRINT "RESTART THE PROGRAM --- PRESS THE SPACE BAR."
00160 PRINT :PRINT "DO YOU WANT INSTRUCTIONS OR DO YOU WANT TO EN
TER"
00170 LINE INPUT "THE PROGRAM ?? (I=INSTRUCTIONS//P=PROGRAM)...";
Q$
00180 IF LEFT$(Q$,1)="P" GOTO 330
00190 PRINT :PRINT
00200 PRINT "THIS PROGRAM, WITH YOUR ASSISTANCE, WILL DESIGN BAND
PASS"
00210 PRINT "AND NOTCH FILTER CIRCUITS USING OPERATIONAL AMPLIFIE
RS."
00220 PRINT "THESE FILTERS ARE VARIABLE GAIN AND, THEREFORE, VARI
ABLE"
00230 PRINT "BANDWIDTH. THE GAIN AND BANDWIDTH ARE CONTROLLED BY
ONE"
00240 PRINT "COMPONENT WHICH WILL BE IDENTIFIED LATER IN THE PROG
RAM."
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: BANDPASS DESIGN - DACEY

```
00250 PRINT
00260 PRINT "YOU WILL BE ASKED THREE THINGS:"
00270 PRINT TAB(5)"1. THE TYPE OF FILTER DESIRED - BANDPASS OR NO
TCH."
00280 PRINT TAB(5)"2. THE DESIRED CENTER FREQUENCY."
00290 PRINT TAB(5)"3. THE MINIMUM INPUT IMPEDANCE."
00300 PRINT :PAUSE
00310 PRINT TAB(5)"***YOUR CHOICE OF CENTER FREQUENCIES SHOULD BE
***"
00320 PRINT TAB(5)"***NO GREATER THAN 500 KHZ FOR THIS PROGRAM.
***"
00330 PRINT :PRINT :PRINT
00340 PRINT "ARE YOU INTERESTED IN DESIGNING A BANDPASS FILTER OR
"
00350 LINE INPUT "A NOTCH FILTER ? (B=BANDPASS//N=NOTCH) ...";Q$
00360 PRINT :PRINT "HERE IS THE SCHEMATIC OF THE CIRCUIT YOU CHOS
E."
00370 PRINT "COPY THE CIRCUIT AND CONNECT ALL THE DOTTED LINES."
00380 IF LEFT$(Q$,1)="B" GOTO 540
00390 IF LEFT$(Q$,1)="N" GOTO 850
00400 GOTO 340
00410 GOSUB 1250
00420 R1=1/(2*P1*F*C1)
00430 R1=INT(R1*10+.5)/10
00440 R2=1/(3*P1*F*C1)
00450 R2=INT(R2*10+.5)/10
00460 GOSUB 1390
00470 IF F$="NOTCH" GOTO 510
00480 PRINT "THE RESISTOR R2 SHOULD BE PARTIALLY VARIABLE. IN THI
S WAY"
00490 PRINT "THE GAIN OF THE PASSBAND AND THE Q CAN BE VARIED."
00500 END
00510 PRINT "THE 50K POT CONTROLS THE Q AND THE GAIN OF THE FILTE
R."
00520 PRINT "THE 0 IN THE SCHEMATIC INDICATES NO CONNECTION."
00530 END
00540 PRINT :PRINT :PRINT TAB(8)"*****";TAB(18);
00550 PRINT "OPERATIONAL AMPLIFIER BANDPASS FILTER *****"
00560 F$="BANDPASS":PRINT :PRINT
00570 PRINT TAB(12)"!";:FOR I=1TO 26:PRINT "-";:NEXT I:PRINT "!";
00580 GOSUB 840:PRINT "[ R1 ]":GOSUB 840
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: BANDPASS DESIGN - DACEY < CONT'D >

```
00590 PRINT "!";GOSUB 840:PRINT "[ R1 ]";GOSUB 840:PRINT "!"
00600 PRINT TAB(12)"!";TAB(39)"!";TAB(52)"!";TAB(65)"!
00610 PRINT TAB(12)"!";TAB(16)"!-";TAB(38);GOSUB 840:PRINT " C1"
;
00620 PRINT TAB(51);GOSUB 840:PRINT " C2";TAB(65)"!
00630 PRINT TAB(12)"!";GOSUB 840:PRINT "!";TAB(18)"-";TAB(22)"-
;
00640 PRINT TAB(38);GOSUB 840:PRINT TAB(51);GOSUB 840:PRINT TAB
(65)"!
00650 PRINT TAB(16)"!";TAB(27)"-";TAB(39)"!";TAB(52)"!";TAB(65)"!
"
00660 PRINT TAB(16)"!";TAB(18)CHR$(34);"741 OP-AMP";CHR$(34);TAB(
32);
00670 GOSUB 840:PRINT "!";TAB(39)"!";GOSUB 840:PRINT "[ R2 ]";
00680 GOSUB 840:PRINT "!";TAB(65)"!";PRINT TAB(6)"! ! C3";TAB(16)
"!";
00690 PRINT TAB(27)"-";TAB(35)"!";TAB(39)"!";TAB(52)"!";TAB(65)"!
"
00700 PRINT TAB(2)"X";GOSUB 840:PRINT "! !";GOSUB 840:PRINT "!"
;
00710 GOSUB 840:PRINT "!";TAB(18)"+";TAB(22)"-";TAB(35)"!";TAB(38
);
00720 GOSUB 840:PRINT " C1";TAB(51)"GND";TAB(65)"!";GOSUB 840
00730 PRINT "X":PAUSE
00740 PRINT " IN";TAB(6)"! !";TAB(12)"!";TAB(16)"!-";TAB(35)"!";
00750 PRINT TAB(38);GOSUB 840:PRINT TAB(65)"!";TAB(67)"OUT"
00760 PRINT TAB(2)"X";TAB(11);GOSUB 840:PRINT TAB(35)"!";
00770 PRINT TAB(39)"!";TAB(65)"!";TAB(69)"X"
00780 PRINT TAB(2)"!";TAB(11)"!R!";TAB(35)"!";GOSUB 840:PRINT "!"
;
00790 FOR I=1TO 25:PRINT "-";NEXT I:PRINT "!";TAB(69)"!"
00800 PRINT TAB(2)"!";TAB(11)"!3!";TAB(69)"!"
00810 PRINT TAB(2)"!";TAB(11);GOSUB 840:PRINT TAB(69)"!"
00820 PRINT TAB(2)"!";TAB(12)"!";TAB(69)"!"
00830 PRINT TAB(1)"GND";TAB(11)"GND";TAB(67)"GND":GOTO 410
00840 PRINT "---";RETURN
00850 PRINT :PRINT :PRINT TAB(8)"*****";TAB(18);
00860 PRINT "OPERATIONAL AMPLIFIER NOTCH FILTER *****"
00870 F$="NOTCH":PRINT :PRINT
00880 PRINT TAB(45)"!";FOR I=1TO 14:PRINT "-";NEXT I:PRINT "!"
00890 PRINT TAB(11)"!";FOR I=1TO 13:PRINT "-";NEXT I:PRINT "!"
```



<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: BANDPASS DESIGN - DACEY < CONT'D >

```
00900 PRINT TAB(45)"!";TAB(49)"!-";TAB(60)"!"
00910 PRINT TAB(11)"!";TAB(15)"!-";TAB(25)"!";TAB(28)"C1";TAB(38)
"C1";
00920 PRINT TAB(45)"!";:GOSUB 1240:PRINT "!-";TAB(54)"-";TAB(60)"
!";
00930 PRINT TAB(4)"C3";TAB(11)"!";:GOSUB 1240:PRINT "!-";TAB(19)"
-";
00940 PRINT TAB(25)"!";TAB(29)"! !";TAB(39)"! !";TAB(49)"!";TAB(5
1);
00950 PRINT CHR$(34);"741";CHR$(34);TAB(57);:GOSUB 1240
00960 PRINT "!";:GOSUB 1240:PRINT "!";:GOSUB 1240:PRINT "X"
00970 PRINT TAB(5)"! !";TAB(15)"!";TAB(17)CHR$(34);"741";CHR$(34)
;
00980 PRINT TAB(22);:GOSUB 1240:PRINT "!";:GOSUB 1240:PRINT "! !";
;
00990 GOSUB 1240:PRINT "!";:GOSUB 1240:PRINT "! !";:GOSUB 1240
01000 PRINT "!";:GOSUB 1240:PRINT TAB(49)"!+";TAB(54)"-";TAB(64)"
! OUT"
01010 PRINT TAB(1)"X";:GOSUB 1240:PRINT "! !";:GOSUB 1240:PRINT "
!";
01020 GOSUB 1240:PRINT "!+";TAB(19)"-";TAB(25)"!";TAB(29)"! !";TA
B(34);
01030 GOSUB 1240:PRINT TAB(39)"! !";TAB(45)"!";TAB(49)"!-";TAB(64)
)"!";
01040 PRINT TAB(68)"X":PRINT TAB(1)"IN";TAB(5)"! !";TAB(11)"!";TA
B(15);
01050 PRINT "!-";TAB(25)"!";TAB(34);"!R!";TAB(45)"!";TAB(63);:GOS
UB 1240
01060 PRINT TAB(68)"!";:PRINT TAB(1)"X";TAB(10);:GOSUB 1240
01070 PRINT TAB(25)"!";TAB(34)"!2!";TAB(45)"!";TAB(58)"-!";TAB(63)
)"!5!";
01080 PRINT TAB(68)"!";:PRINT TAB(1)"!";TAB(10)"!R!";TAB(25)"!";TA
B(34);
01090 GOSUB 1240:PRINT TAB(45)"!";TAB(55)"-";TAB(58)"!";:GOSUB 1
240
01100 PRINT "!0!";TAB(68)"!";:PAUSE :PRINT
01110 PRINT TAB(1)"!";TAB(10)"!3!";TAB(25)"!";TAB(35)"!";:GOSUB 1
240
01120 GOSUB 1240:GOSUB 1240:PRINT "0";:GOSUB 1240:PRINT "!";:GOSU
B 1240
01130 PRINT TAB(53)CHR$(34);"741";CHR$(34);TAB(59)"!";TAB(63)"!K!
";
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: BANDPASS DESIGN - DACEY

```
01140 PRINT TAB(68)"!":PRINT TAB(1)"!";TAB(10);:GOSUB 1240
01150 PRINT TAB(25)"!";TAB(30)"C2";TAB(34);:GOSUB 1240:PRINT TAB(
45);
01160 PRINT "!";TAB(49)"!";TAB(55)"-";TAB(58)"-!-! ---";TAB(68)"!
"
01170 PRINT TAB(1)"!";TAB(11)"!";TAB(25)"!";TAB(34);:GOSUB 1240
01180 PRINT TAB(45)"!";TAB(49)"!";TAB(58)"-! !";TAB(64)"!";TAB(68
)"!";
01190 PRINT TAB(1)"!";TAB(11)"!";TAB(25)"!";TAB(35)"!";TAB(45)"!";
;
01200 PRINT TAB(49)"!";:FOR I=1TO 11:PRINT "-";:NEXT I:PRINT "!";
01210 PRINT TAB(64)"!";TAB(68)"!";
01220 PRINT " GND";TAB(10)"GND";TAB(25)"!-[ R1 ]-!-[ R1 ]-!";
01230 PRINT TAB(63)"GND";TAB(67)"GND":GOTO 410
01240 PRINT "----";:RETURN
01250 PRINT :PRINT :F=0:R=0
01260 INPUT "SPECIFY THE DESIRED CENTER FREQUENCY IN KILOHERTZ.
";F
01270 PRINT
01280 INPUT "SPECIFY THE MINIMUM INPUT IMPEDANCE IN K-OHMS.
";R
01290 PRINT :PRINT
01300 PRINT "HERE ARE THE COMPONENT VALUES FOR THE ";F;" FILTER
WHEN"
01310 PRINT "THE CENTER FREQUENCY IS ";F;" KHZ."
01320 IF F<10 THEN C1=.01:C1$=" 0.01"+L1$:C2$=" 0.02"+L1$:GOTO 13
50
01330 IF F>100 THEN C1=.0001:C1$=" 100 "+L2$:C2$=" 200 "+L2$:GOTO
1350
01340 C1=.001:C1$=" 0.001"+L1$:C2$=" 0.002"+L1$
01350 R3=R:C3=INT(10*M/(P1*F*R))
01360 IF C3<10000 THEN C3$=STR$(C3):C3$=C3$+L2$
01370 IF C3>=10000 THEN C3=INT(C3/N):GOSUB 1480
01380 RETURN
01390 PRINT :PRINT :L3$=" K OHMS"
01400 PRINT "THE VALUE OF R1 IS ";R1;L3$
01410 PRINT "THE VALUE OF R2 IS ";R2;L3$
01420 PRINT "THE VALUE OF R3 IS ";R3;L3$
01430 PRINT "THE VALUE OF C1 IS ";C1$
01440 PRINT "THE VALUE OF C2 IS ";C2$
01450 PRINT "THE VALUE OF C3 IS ";C3$
```

<H><U><G>    <S><O><F><T><W><A><R><E>    <U><O><L>    II

PROGRAM NAME: BANDPASS DESIGN - DACEY &lt; CONT'D &gt;

```

01460 PRINT :PAUSE
01470 RETURN
01480 C3$="":C4$="":C5$=""
01490 C4$=STR$(C3)
01500 X=LEN(C4$):X1=X-1
01510 IF C3<10 THEN C3$=" 0.0"
01520 IF C3>=10 THEN C3$=" 0.":IF C3>=100 GOTO 1590
01530 FOR I=2TO X1
01540 C5$=MID$(C4$,I,1)
01550 C3$=C3$+C5$
01560 NEXT I
01570 C3$=C3$+L1$
01580 RETURN
01590 C3$="":FOR I=1TO 2
01600 C5$=MID$(C4$,I,1)
01610 C3$=C3$+C5$
01620 NEXT I
01630 C3$=C3$+L1$
01640 RETURN

```

11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: AMPLIFIER DESIGN - DACEY

```
00010 REM ////////////////////////////////////////////
//////
00020 REM >>>>>> OPERATIONAL AMPLIFIER DESIGN PROGRAM <<
<<<<<
00030 REM >>>>>> COPYRIGHT 1978 FRANK DACEY <<
<<<<<
00040 REM ////////////////////////////////////////////
//////
00050 REM ***** BY : FRANK DACEY **
*****
00060 REM ***** 328 BARBARA DRIVE **
*****
00070 REM ***** POINT PLEASANT, NJ 08742 **
*****
00080 REM ////////////////////////////////////////////
//////
00090 C=0:L2$=" K OHMS":L3$=" K OHMS":F2$=" KHZ"
00100 FOR I=1 TO 4:PRINT :NEXT I:PRINT TAB(5)"*****";TAB(18);
00110 PRINT "OPERATIONAL AMPLIFIER DESIGN PROGRAM *****":P
RINT
00120 PRINT :PRINT "DURING THE PROGRAM THERE WILL BE OCCASIONAL P
AUSES"
00130 PRINT "TO ALLOW TIME FOR READING AND/OR DRAWING SCHEMATICS.
TO"
00140 PRINT "RESTART THE PROGRAM --- PRESS THE SPACE BAR."
00150 PRINT :PRINT "DO YOU WANT INSTRUCTIONS OR DO YOU WANT TO EN
TER"
00160 LINE INPUT "THE PROGRAM ? (I=INSTRUCTIONS/P=PROGRAM)...":Q$
00170 IF LEFT$(Q$,1)="I" GOTO 200
00180 IF LEFT$(Q$,1)="P" GOTO 440
00190 PRINT :PRINT "LET'S TRY THAT AGAIN":GOTO 150
00200 PRINT :PRINT
00210 PRINT "THIS PROGRAM, WITH YOUR ASSISTANCE, WILL DESIGN THRE
E TYPES"
00220 PRINT "OF AMPLIFIERS. AN INVERTING AMPLIFIER, WHERE THE OUT
PUT IS"
00230 PRINT "180 DEGREES OUT OF PHASE WITH THE INPUT, A NON-INVER
TING"
00240 PRINT "AMPLIFIER, WHERE THE OUTPUT IS IN PHASE WITH THE INP
UT, AND"
00250 PRINT "A DIFFERENTIAL AMPLIFIER. THE DIFFERENTIAL AMPLIFIER
HAS AN"
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: AMPLIFIER DESIGN - DACEY

```
00260 PRINT "INVERTED OUTPUT ALSO."
00270 PRINT :PRINT "YOU WILL BE ASKED FOUR THINGS:"
00280 PRINT TAB(5)"1. THE TYPE OF AMPLIFIER YOU WANT TO DESIGN."
00290 PRINT TAB(5)"2. THE AMOUNT OF VOLTAGE GAIN YOU DESIRE."
00300 PRINT TAB(5)"3. THE MINIMUM INPUT IMPEDANCE."
00310 PRINT TAB(5)"4. THE HIGHEST FREQUENCY YOU WANT AMPLIFIED. ";
:PAUSE
00320 PRINT :PRINT
00330 PRINT "THE PROGRAM WILL:"
00340 PRINT TAB(5)"1. DRAW A SCHEMATIC OF THE CIRCUIT YOU CHOOSE.
"
00350 PRINT TAB(5)"2. PRINT A LISTING OF ALL COMPONENT VALUES."
00360 PRINT TAB(5)"3. STATE THE FREQUENCY WHERE THE VOLTAGE GAIN"
00370 PRINT TAB(5)"IS DOWN THREE DB. (AMPLIFIER BANDWIDTH)"
00380 PRINT TAB(5)"4. STATE THE VOLTAGE GAIN IN DECIBELS.":PRINT
00390 PRINT TAB(5)"*** THIS PROGRAM USES THE TYPE 741 OP-AMP FOR
***"
00400 PRINT TAB(5)"*** ALL AMPLIFIERS. THE GAIN BANDWIDTH PRODUCT
***"
00410 PRINT TAB(5)"*** IS 1 MHZ. THEREFORE, THE GAIN TIMES BAND-
***"
00420 PRINT TAB(5)"*** WIDTH IS ALWAYS 1 MHZ FOR THIS OP-AMP.
***"
00430 PAUSE
00440 PRINT
00450 PRINT
00460 PRINT
00470 PRINT "WHAT TYPE OF AMPLIFIER DO YOU WANT TO DESIGN ?"
00480 PRINT
00490 PRINT " 1. AN INVERTING AMPLIFIER."
00500 PRINT " 2. A NON-INVERTING AMPLIFIER."
00510 PRINT " 3. A DIFFERENTIAL AMPLIFIER."
00520 PRINT " 0. END OF PROGRAM PLEASE - I'M FINISHED."
00530 PRINT :PRINT
00540 PRINT "INDICATE YOUR PREFERENCE BY USING THE NUMBER THAT"
00550 INPUT "PRECEDES YOUR CHOICE. " ;A
00560 PRINT :PRINT
00570 IF A=0 GOTO 620
00580 IF A<4 GOTO 600
00590 PRINT :PRINT "OOPS":GOTO 540
00600 ON A GOSUB 640,710,800
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: AMPLIFIER DESIGN - DACEY < CONT'D >

```
00610 GOTO 440
00620 PRINT TAB(20)"***** END OF PROGRAM *****"
00630 END
00640 A$="INVERTING AMPLIFIER"
00650 GOSUB 880
00660 GOSUB 900
00670 GOSUB 990
00680 GOSUB 1200
00690 GOSUB 1330
00700 RETURN
00710 A$="NON-INVERTING AMPLIFIER"
00720 GOSUB 880
00730 GOSUB 900
00740 GOSUB 1060
00750 GOSUB 1200
00760 GOSUB 1330
00770 PRINT "R2 SHOULD BE REDUCED BY AN AMOUNT EQUAL TO THE INTER
NAL"
00780 PRINT "RESISTANCE OF THE GENERATOR.":PAUSE
00790 RETURN
00800 A$="DIFFERENTIAL AMPLIFIER"
00810 GOSUB 880
00820 GOSUB 900
00830 GOSUB 1110
00840 GOSUB 1200
00850 C=1
00860 GOSUB 1330
00870 RETURN
00880 PRINT TAB(8)"*** HERE IS THE SCHEMATIC OF THE ";A$;" ***"
00890 RETURN
00900 PRINT
00910 PRINT TAB(35)"!";GOSUB 1190:PRINT "[ R3 ]";GOSUB 1190:PRI
NT "--!"
00920 PRINT TAB(35)"!";TAB(50)"!"
00930 PRINT TAB(35)"!";TAB(39)"!-";TAB(50)"!"
00940 PRINT TAB(13)"!";FOR I=1TO 12:PRINT "-";NEXT I:PRINT "[ R
1 ]";
00950 GOSUB 1190:PRINT "!";GOSUB 1190:PRINT "!-";TAB(44)"-";TAB(
50)"!"
00960 PRINT TAB(13)"!";TAB(39)"!";" ";CHR$(34);"741";CHR$(34);TAB
(47);
00970 GOSUB 1190:PRINT "!";GOSUB 1190:PRINT "X"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: AMPLIFIER DESIGN - DACEY < CONT'D >

```
00980 RETURN
00990 PRINT TAB(13)"X";TAB(23)"!";:GOSUB 1190:GOSUB 1190:PRINT "[
R2 ]";
01000 GOSUB 1190:PRINT "!+";TAB(44)"-";TAB(53)"OUT"
01010 PRINT TAB(12)"IN";TAB(23)"!";TAB(39)"!-";TAB(54)"X"
01020 PRINT TAB(13)"X";TAB(23)"!";TAB(54)"!"
01030 PRINT TAB(13)"!";TAB(23)"!";TAB(54)"!"
01040 PRINT TAB(12)"GND";TAB(22)"GND";TAB(53)"GND";:PAUSE
01050 RETURN
01060 PRINT TAB(13)"!";TAB(23)"!";:GOSUB 1190:GOSUB 1190:PRINT "[
R2 ]";
01070 GOSUB 1190:PRINT "!+";TAB(44)"-";TAB(53)"OUT"
01080 PRINT TAB(13)"!";TAB(23)"X";TAB(39)"!-";TAB(54)"!"
01090 PRINT TAB(13)"!";TAB(22)"IN";TAB(54)"!"
01100 PRINT TAB(13)"!";TAB(23)"X";TAB(54)"!";GOTO 1040
01110 PRINT TAB(13)"!";TAB(22)"!";:GOSUB 1190:PRINT "[ R1 ]";
01120 GOSUB 1190:PRINT "!";:GOSUB 1190:PRINT "!+";TAB(44)"-";TAB(
53)"OUT"
01130 PRINT TAB(13)"X";TAB(22)"X";TAB(35)"!";TAB(39)"!-";TAB(54)"
X"
01140 PRINT TAB(10)"(IN#1)";TAB(19)"(IN#2)";TAB(35)"!";TAB(54)"!"
01150 PRINT TAB(13)"X";TAB(22)"X";TAB(35)"!";:GOSUB 1190:PRINT "[
R3 ]";
01160 GOSUB 1190:GOSUB 1190:GOSUB 1190:PRINT "!"
01170 PRINT TAB(12)"GND";TAB(21)"GND";TAB(53)"GND";:PAUSE
01180 RETURN
01190 PRINT "---";:RETURN
01200 PRINT :PRINT :PRINT "HOW MUCH VOLTAGE GAIN DO YOU WANT THE
";A$
01210 INPUT "TO HAVE ? (ANYTHING FROM 1 TO 10000)..... ";B
01220 PRINT :PRINT "STATE THE MINIMUM INPUT IMPEDANCE YOU WANT TH
E ";A$
01230 INPUT "TO HAVE . (IN K OHMS)..... ";Z
01240 PRINT :PRINT "STATE THE HIGHEST FREQUENCY YOU WANT THE ";A$
01250 INPUT "TO AMPLIFY. (IN KHZ)..... ";F
01260 F2=1000/B:IF F2<1 THEN F2=F2*1000:F2$=" HZ":GOTO 1280
01270 F2=INT(10*F2+.5)/10:IF F2<F GOTO 1290
01280 RETURN
01290 PRINT :PRINT "YOUR 3 DB DOWN FREQUENCY IS LESS THAN THE HIG
HEST"
01300 PRINT "FREQUENCY YOU WANT TO AMPLIFY. YOU MUST USE LESS"
```







<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RC OSC DESIGN - DACEY

```
00010 REM ////////////////////////////////////////////
//////
00020 REM >>>>>>          R-C PHASE SHIFT OSCILLATOR          <<
<<<<<<
00030 REM >>>>>>          COPYRIGHT 1978 FRANK DACEY          <<
<<<<<<
00040 REM ////////////////////////////////////////////
//////
00050 REM *****      BY :      FRANK DACEY                  **
*****
00060 REM *****      328 BARBARA DRIVE                      **
*****
00070 REM *****      POINT PLEASANT, NJ 08742              **
*****
00080 REM ////////////////////////////////////////////
//////
00090 P1=3.14159:M=1000000:K=1000:L1$=" MICROFARAD":L3$="":C$=""
00100 FOR I=1 TO 6:PRINT :NEXT I:PRINT TAB(5)"*****";TAB(15);
00110 PRINT "R-C PHASE SHIFT OSCILLATOR DESIGN PROGRAM          *****
"
00120 PRINT :PRINT "DURING THE PROGRAM THERE WILL BE OCCASIONAL P
AUSES"
00130 PRINT "TO ALLOW TIME FOR READING AND/OR DRAWING SCHEMATICS.
TO"
00140 PRINT "RESTART THE PROGRAM --- PRESS THE SPACE BAR."
00150 PRINT :PRINT "DO YOU WANT INSTRUCTIONS OR DO YOU WANT TO EN
TER"
00160 LINE INPUT "THE PROGRAM ? (I=INSTRUCTIONS/P=PROGRAM)      ";Q$
00170 IF LEFT$(Q$,1)="P" GOTO 410
00180 PRINT :PRINT
00190 PRINT "THIS PROGRAM WILL, WITH YOUR ASSISTANCE, DESIGN R-C
PHASE"
00200 PRINT "SHIFT OSCILLATORS USING THE TYPE 741 OPERATIONAL AMP
LIFIER."
00210 PRINT "FOR BEST RESULTS USE GOOD QUALITY FILM CAPACITORS AN
D LOW"
00220 PRINT "TOLERANCE FILM RESISTORS. ONE OF THE RESISTORS SHOUL
D BE"
00230 PRINT "MADE VARIABLE SO SMALL FREQUENCY CHANGES CAN BE MADE
"
00240 PRINT :PRINT TAB(5)"THE PROGRAM WILL:"
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RC OSC DESIGN - DACEY < CONT'D >

```
00250 PRINT TAB(15)"1. DRAW THE CIRCUIT SCHEMATIC."
00260 PRINT TAB(15)"2. LIST COMPONENT VALUES FOR THE CHOSEN FREQU
ENCY."
00270 PRINT
00280 PAUSE
00290 PRINT TAB(5)***REMEMBER - YOU MUST ALSO CONNECT PLUS AND M
INUS***
00300 PRINT TAB(5)***POWER SUPPLY VOLTAGES IN ADDITION TO THE OT
HER ***
00310 PRINT TAB(5)***COMPONENTS SHOWN IN THE SCHEMATIC. ANY VOL
TAGE***
00320 PRINT TAB(5)***BETWEEN ABOUT EIGHT TO FIFTEEN VOLTS WILL W
ORK.***
00330 PRINT :PRINT TAB(5)*** PINOUTS FOR THE 741 OP-AMP ***
00340 PRINT TAB(5)** PIN 4 MINUS INPUT * PIN 5 PLUS INPUT * PIN
6- **
00350 PRINT TAB(5)** MINUS SUPPLY VOLTAGE * PIN 10 OUTPUT * PIN
11-**
00360 PRINT TAB(5)** POSITIVE SUPPLY VOLTAGE. (FOR 14 PIN DIP)
**
00370 PRINT
00380 PRINT TAB(5)***YOUR CHOICE OF FREQUENCIES SHOULD BE LIMITE
D***
00390 PRINT TAB(5)***TO 100 KILOHERTZ OR BELOW FOR THIS PROGRAM.
***";
00400 PAUSE :Z=0:PRINT
00410 PRINT :PRINT
00420 PRINT "SPECIFY THE FREQUENCY, IN HERTZ, YOU WANT THE R-C PH
ASE"
00430 INPUT "SHIFT OSCILLATOR TO GENERATE. (FREQUENCY IN HERTZ)
";F
00440 IF F<100 THEN C=1:C$="1.0":GOTO 490
00450 IF F<1000 THEN C=.1:C$="0.1":GOTO 490
00460 IF F<10000 THEN C=.01:C$="0.01":GOTO 490
00470 IF F<=100000 THEN C=.001:C$="0.001":GOTO 490
00480 IF F>100000 GOTO 420
00490 R=M/(2*P1*2.45*F*C):L3$="K OHMS"
00500 IF R<1000 THEN R=INT(R):L3$="OHMS"
00510 IF R>=1000 THEN R=R/K:R=INT(10*R+.5)/10
00520 R4$=" 620 OHMS":R5$=" 18 K OHMS"
00530 PRINT :IF Z=1 GOTO 840
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RC OSC DESIGN - DACEY < CONT'D >

```
00540 PRINT TAB(8)"HERE IS THE R-C PHASE SHIFT OSCILLATOR SCHEMAT
IC"
00550 PRINT :PRINT TAB(5)"!-";:GOSUB 830:PRINT "[ R5 ]---!"
00560 PRINT TAB(5)"!";TAB(19)"!"
00570 PRINT TAB(5)"!";TAB(14)"-!";TAB(19)"!"
00580 PRINT TAB(5)"!";TAB(11)"-";TAB(14)"-!";:GOSUB 830:PRINT "!"
;
00590 GOSUB 830:PRINT "[ R4 ]";:FOR I=1TO 15:PRINT "-";:NEXT I:PR
INT "!"
00600 PRINT TAB(5)"!--- "CHR$(34)"741"CHR$(34)"!";TAB(44)"!"
00610 PRINT TAB(5)"!";TAB(11)"-";TAB(14)"+"!-";:GOSUB 830:GOSUB 83
0
00620 PRINT "[ R6 ]";:GOSUB 830:PRINT "!";TAB(44)"!";
00630 FOR I=1TO 13:PRINT "-";:NEXT I:PRINT "!"
00640 PRINT TAB(5)"!";TAB(14)"-!";TAB(32)"!";TAB(44)"!";TAB(58)"!
"
00650 PRINT TAB(5)"!";TAB(31)"GND";TAB(44)"!";TAB(48)"!-";TAB(58)
"!";
00660 PRINT TAB(5)"!";TAB(11)"C1";TAB(23)"C2";TAB(38)"C3";TAB(44)
"!";
00670 GOSUB 830:PRINT "!--";TAB(52)"-";TAB(58)"!";:PAUSE
00680 PRINT :PRINT TAB(5)"!";TAB(9)"! !";TAB(21)"! !";TAB(36)"! !
";
00690 PRINT TAB(48)"! ";CHR$(34)"741"CHR$(34);:GOSUB 830:PRINT "!"
"
00700 PRINT TAB(5)"!";:GOSUB 830:PRINT "! !";:GOSUB 830:PRINT "!--
-";
00710 GOSUB 830:PRINT "! !";:GOSUB 830:GOSUB 830:PRINT "!---";
00720 GOSUB 830:PRINT "! !";:GOSUB 830:PRINT "--!";
00730 GOSUB 830:PRINT "!!+";TAB(52)"-";TAB(58)"X"
00740 PRINT TAB(9)"! !";TAB(15)"!";TAB(21)"! !";TAB(30)"!";
00750 PRINT TAB(36)"! !";TAB(44)"!";TAB(48)"!-";TAB(57)"OUT"
00760 PRINT TAB(15)"!";TAB(30)"!";TAB(44)"!";TAB(58)"!"
00770 PRINT TAB(15)"!-[ R1 ]-!";TAB(30)"!-[ R2 ]-!";
00780 PRINT TAB(44)"!-[ R3 ]-!";TAB(58)"!"
00790 PRINT TAB(24)"!";TAB(39)"!";TAB(53)"!";TAB(58)"!"
00800 PRINT TAB(23)"GND";TAB(38)"GND";TAB(52)"GND";TAB(57)"GND"
00810 PRINT :PRINT TAB(10)"USING A PENCIL, TRACE OVER ALL DASHED
LINES."
00820 PAUSE :PRINT :PRINT :GOTO 840
00830 PRINT "---";:RETURN
```







<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DECOCT - KERN

```
01000 REM DECOCT
01010 CNTRL 0.9000
01020 PRINT
01030 INPUT "APPROXIMATE NUMBER OF BYTES IN PROGRAM TO BE ENTERED
: ";B
01040 DIM A(B),D(B)
01050 LINE INPUT "CHOOSE INPUT METHOD ('KEYBOARD' OR 'BINARY TAPE
')";I$
01060 IF ASC(I$)=66 OR ASC(I$)=75 THEN 1090
01070 PRINT "YOU MUST TYPE 'KEYBOARD' OR 'BINARY TAPE.'"
01080 GOTO 1050
01090 IF ASC(I$) <> 75 THEN 1110
01100 GOTO 5000
01110 IF ASC(I$) <> 66 THEN 3000
01120 GOTO 6000
03000 PRINT
03010 CLEAR N3: CLEAR N4: CLEAR N5
03020 LINE INPUT "WANT PROGRAM DISPLAYED (YES OR NO)? ";Q3$
03030 IF Q3$ < "Y" THEN 4000
03040 LINE INPUT "WANT ADDRESSES DISPLAYED IN DECIMAL OR OCTAL? "
;Q4$
03050 LINE INPUT "WANT DATA DISPLAYED IN DECIMAL OR OCTAL? ";Q5$
03060 PRINT "TYPE A 'RETURN' TO SCROLL DISPLAY. TYPE A 'CONTROL-
B'"
03070 PRINT "TO CORRECT AN ERROR:"
03080 PRINT
03090 PRINT TAB(15);"ADDRESS";TAB(25);"DATA"
03100 FOR J=0 TO I-1
03110 N=A(J)
03115 PRINT TAB(15);
03120 IF Q4$ < "O" THEN 3180
03130 GOSUB 8000
03140 IF INT(LOG(N)/LOG(10)) = 5 THEN 3180
03150 FOR L=1 TO 5-INT(LOG(N)/LOG(10))
03160 PRINT "0";
03170 NEXT
03180 PRINT MID$(STR$(N),2);TAB(25);
03190 N=D(J)
03200 IF Q5$ < "O" THEN 3300
03210 CLEAR N3: CLEAR N4: CLEAR N5
03220 GOSUB 8500
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DECOCT - KERN < CONT'D >

```
03230 O=2
03240 IF N=0 THEN 3270
03250 O=2-INT(LOG(N)/LOG(10))
03260 IF O=0 THEN 3300
03270 FOR L=1 TO O
03280 PRINT "0";
03290 NEXT
03300 PRINT MID$(STR$(N),2);
03310 IF K=7 THEN 3330
03320 IF (K-7)/12 <> INT((K-7)/12) THEN 3340
03330 PAUSE
03340 K=K+1
03350 PRINT
03360 NEXT J
04000 PRINT
04010 LINE INPUT "WANT TO PUT DECIMAL DATA ON MAG TAPE (YES OR NO
)? ";Q6$
04020 IF Q6$ < "Y" THEN 4270
04030 CLEAR J: CLEAR K: CLEAR L: CLEAR O: CLEAR X
04040 CLEAR N: CLEAR N1: CLEAR N2: CLEAR N3: CLEAR N4: CLEAR N5
04050 Q0$="": Q1$="": Q2$="": Q3$="": Q4$="": Q5$="": Q6$=""
04060 Q7$="": Q8$="": Q9$="": I$="": N$="": D$=""
04070 CLEAR Q0$: CLEAR Q1$: CLEAR Q2$: CLEAR Q3$: CLEAR Q4$: CLEAR Q5$
: CLEAR Q6$
04080 CLEAR Q7$: CLEAR Q8$: CLEAR Q9$: CLEAR I$: CLEAR N$: CLEAR D$
04090 S=A(0)
04100 E=A(I-1)
04110 CLEAR A(
04120 FOR I=E-S+1 TO E
04130 D(I)=0
04140 NEXT
04150 CLEAR B: CLEAR I
04160 PRINT
04170 PRINT "DECIMAL DATA FORMAT:"
04180 PRINT
04190 PRINT "STARTING ADDRESS = S";SPC(10);
04200 PRINT "ENDING ADDRESS = E"
04210 PRINT "PROGRAM BYTES = D("
04220 PRINT "    -- FIRST BYTE = D(0)"
04230 PRINT "    -- LAST BYTE = D(E-S)"
04240 PRINT
```

<H><U><G> <S><O><F><T><U><A><R><E> <U><O><L> II

PROGRAM NAME: DECOCT - KERN < CONT'D >

```
04250 PRINT "WHEN PROGRAM ENDS, USE 'PUT' COMMAND TO RECORD"
04260 PRINT "DECIMAL DATA ON MAG TAPE."
04270 END
05000 REM KEYBOARD ENTRY SUBROUTINE
05010 LINE INPUT "WANT TO ENTER STARTING ADDRESS IN DECIMAL OR OC
TAL? ";Q0$
05020 LINE INPUT "STARTING ADDRESS? ";Q1$
05030 IF Q0$ < "0" THEN 5100
05040 FOR I=1 TO LEN(Q1$)
05050 IF MID$(Q1$,I,1) = CHR$(32) THEN 5070
05060 N$ = N$ + MID$(Q1$,I,1)
05070 NEXT
05080 Q1$ = N$
05090 CLEAR I
05100 N=VAL(Q1$)
05110 IF Q0$ < "0" THEN 5130
05120 GOSUB 7000
05130 A(I)=N
05140 PRINT
05150 LINE INPUT "WANT TO ENTER DATA IN DECIMAL OR OCTAL? ";Q2$
05160 PRINT
05170 PRINT "ENTER DATA ONE BYTE AT A TIME. ENTER 'FIX' TO"
05180 PRINT "CORRECT AN ERROR. ENTER 'END' WHEN FINISHED."
05190 PRINT
05200 PRINT "ADDRESS";TAB(10);"DATA"
05205 PRINT
05210 IF Q0$ < "0" THEN 5270
05220 GOSUB 8000
05230 IF INT(LOG(N)/LOG(10)) = 5 THEN 5270
05240 FOR J=1 TO 5-INT(LOG(N)/LOG(10))
05250 PRINT "0";
05260 NEXT
05270 PRINT MID$(STR$(N),2);TAB(10);
05280 LINE INPUT " ";D$
05290 IF D$ >= "F" THEN 9000
05300 IF D$ < CHR$(58) THEN 5310
05305 GOTO 3000
05310 N=VAL(D$)
05320 IF Q2$ < "0" THEN 5370
05330 IF VAL(MID$(D$,1,1)) > 3 THEN 5900
05340 IF VAL(MID$(D$,2,1)) > 7 OR VAL(MID$(D$,3,1)) > 7 THEN 5900
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: DECOCT - KERN < CONT'D >

```
05350 CLEAR N3: CLEAR N4: CLEAR N5
05360 GOSUB 7500
05370 D(I)=N
05380 A(I+1) = A(I)+1
05390 I=I+1
05400 N=A(I)
05410 GOTO 5210
05900 PRINT "DATA NOT IN OCTAL FORMAT. RE-ENTER BYTE."
05910 PRINT TAB(10);
05920 GOTO 5280
06000 REM BINARY TAPE SUBROUTINE
06010 REM
06020 REM WRITE CALL TO PAM 'RMEM' ROUTINE INTO MEMORY
06030 POKE 28668,205
06040 POKE 28669,177
06050 POKE 28670,1
06060 POKE 28671,201
06070 REM CHANGE ADDRESS OF 'USRFN' TO 28668(D)
06080 POKE 17990,252
06090 POKE 17991,111
06100 PRINT "PLACE TAPE IN MACHINE. TYPE A 'RETURN' WHEN READY."
06110 PAUSE
06120 CNTRL 2,2
06130 X=USR(0)
06140 CNTRL 2,0
06150 PRINT
06155 PRINT CHR$(7);"BINARY TAPE OKAY."
06160 FOR I=0 TO B
06170 IF PEEK(I+27648)=199 THEN 6210
06180 A(I)=I+27648
06190 D(I)=PEEK(I+27648)
06200 NEXT
06210 REM RETURN ADDRESS OF 'USRFN' TO 27648(D)
06220 POKE 17990,0
06230 POKE 17991,108
06240 GOTO 3000
07000 REM OCTAL-TO-DECIMAL SUBROUTINE
07010 IF N <= 377377 THEN 7050
07020 PRINT "NUMBER EXCEEDS 16 BITS."
07030 GOTO 9000
07040 REM HIGH 3 DIGITS
```



<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: DECOCT - KERN < CONT'D >

```
07050 N5 = INT(N/100000)
07060 N = N - N5*100000
07070 N4 = INT(N/10000)
07080 N = N - N4*10000
07090 N3 = INT(N/1000)
07100 N = N - N3*1000
07430 REM LOW 3 DIGITS
07500 IF N <= 377 THEN 7540
07510 PRINT "NUMBER (OR OFFSET-OCTAL SEGMENT) EXCEEDS 8 BITS."
07520 GOTO 9000
07540 N2 = INT(N/100)
07550 N = N - N2*100
07560 N1 = INT(N/10)
07570 N = N - N1*10
07580 N = N5*16384 + N4*2048 + N3*256 + N2*64 + N1*8 + N
07590 RETURN
08000 REM DECIMAL-TO-OCTAL SUBROUTINE
08010 IF N <= 65535 THEN 8050
08020 PRINT "NUMBER EXCEEDS 16 BITS."
08030 GOTO 9000
08040 REM 256-65535 DECIMAL
08050 N5 = INT(N/16384)
08060 N = N - N5*16384
08070 N4 = INT(N/2048)
08080 N = N - N4*2048
08090 N3 = INT(N/256)
08100 N = N - N3*256
08500 IF N <= 255 THEN 8540
08510 PRINT "NUMBER EXCEEDS 8 BITS."
08520 GOTO 9000
08530 REM 0-255 DECIMAL
08540 N2 = INT(N/64)
08550 N = N - N2*64
08560 N1 = INT(N/8)
08570 N = N - N1*8
08580 N = N5*100000 + N4*10000 + N3*1000 + N2*100 + N1*10 + N
08590 RETURN
09000 REM ERROR SUBROUTINE
09010 PRINT "NOTE ADDRESS OF ERROR. TYPE 'RETURN' TO CONTINUE."
09020 PAUSE
09030 PRINT "WANT TO RE-ENTER DATA FROM THIS POINT (TYPE 'RE-ENTE
R') OR"
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> 11

PROGRAM NAME: MORSCII - KERN

```
01000 REM MORSCII
01010 REM
01020 CNTRL 4.1
01030 PRINT
01040 PRINT "THIS PROGRAM TAKES ASCII CHARACTERS THAT ARE ENTERED
FROM THE"
01050 PRINT "KEYBOARD, ENCODES THEM, AND OUTPUTS THEM AS MORSE CO
DE ON THE"
01060 PRINT "H8'S AUDIO OSCILLATOR. ENTER THE MESSAGE ONE LINE A
T A TIME."
01070 PRINT "IF YOU WISH TO RUN LETTERS TOGETHER WITHOUT INTERVEN
ING SPACES,"
01080 PRINT "ENCLOSE THEM IN BRACKETS ( [ ] ). TO SIGNAL THE END
OF A"
01090 PRINT "MESSAGE, TYPE AN ASTERISK (*)."
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
03400 GOTO 1330
03500 GOTO 1330
03600 GOTO 1330
03700 GOTO 1330
03800 GOTO 1330
03900 GOTO 1330
04000 REM '(< OR >)' .....
04005 PAUSE 5*T1
04010 X=USR(3*T)
04015 PAUSE T1
04020 X=USR(T)
04025 PAUSE T1
04030 X=USR(3*T)
04035 PAUSE T1
04040 X=USR(3*T)
04045 PAUSE T1
04050 X=USR(T)
04055 PAUSE T1
04060 X=USR(3*T)
04065 PAUSE T1
04070 GOTO 1330
04100 GOTO 4000
04200 REM '*' (END OF TRANSMISSION) .....
04205 PAUSE 5*T1
04210 X=USR(T)
04215 PAUSE T1
04220 X=USR(3*T)
04225 PAUSE T1
04230 X=USR(T)
04235 PAUSE T1
04240 X=USR(3*T)
04245 PAUSE T1
04250 X=USR(T)
04255 PAUSE T1
04260 GOTO 9999
04300 GOTO 1330
04400 PAUSE 5*T1
04405 X=USR(3*T)
04410 PAUSE T1
04415 X=USR(3*T)
04420 PAUSE T1
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
04425 X=USR(T)
04430 PAUSE T1
04435 X=USR(T)
04440 PAUSE T1
04445 X=USR(3*T)
04450 PAUSE T1
04455 X=USR(3*T)
04460 PAUSE T1
04465 GOTO 1330
04500 REM '/-' .....
04505 PAUSE 5*T1
04510 X=USR(3*T)
04515 PAUSE T1
04520 X=USR(T)
04525 PAUSE T1
04530 X=USR(T)
04535 PAUSE T1
04540 X=USR(T)
04545 PAUSE T1
04550 X=USR(3*T)
04555 PAUSE T1
04560 GOTO 1330
04600 REM '/.' .....
04605 PAUSE 5*T1
04610 X=USR(T)
04615 PAUSE T1
04620 X=USR(3*T)
04625 PAUSE T1
04630 X=USR(T)
04635 PAUSE T1
04640 X=USR(3*T)
04645 PAUSE T1
04650 X=USR(T)
04655 PAUSE T1
04660 X=USR(3*T)
04665 PAUSE T1
04670 GOTO 1330
04700 REM '//' .....
04705 PAUSE 5*T1
04710 X=USR(3*T)
04715 PAUSE T1
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
04720 X=USR(T)
04725 PAUSE T1
04730 X=USR(T)
04735 PAUSE T1
04740 X=USR(3*T)
04745 PAUSE T1
04750 X=USR(T)
04755 PAUSE T1
04760 GOTO 1330
04800 REM '0' -----
04805 X=USR(3*T)
04810 PAUSE T1
04815 X=USR(3*T)
04820 PAUSE T1
04825 X=USR(3*T)
04830 PAUSE T1
04835 X=USR(3*T)
04840 PAUSE T1
04845 X=USR(3*T)
04850 PAUSE T1
04855 GOTO 1330
04900 REM '1' .-----
04905 X=USR(T)
04910 PAUSE T1
04915 X=USR(3*T)
04920 PAUSE T1
04925 X=USR(3*T)
04930 PAUSE T1
04935 X=USR(3*T)
04940 PAUSE T1
04945 X=USR(3*T)
04950 PAUSE T1
04955 GOTO 1330
05000 REM '2' ..-----
05005 X=USR(T)
05010 PAUSE T1
05015 X=USR(T)
05020 PAUSE T1
05025 X=USR(3*T)
05030 PAUSE T1
05035 X=USR(3*T)
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
05040 PAUSE T1
05045 X=USR(3*T)
05050 PAUSE T1
05055 GOTO 1330
05100 REM '3' .....
05105 X=USR(T)
05110 PAUSE T1
05115 X=USR(T)
05120 PAUSE T1
05125 X=USR(T)
05130 PAUSE T1
05135 X=USR(3*T)
05140 PAUSE T1
05145 X=USR(3*T)
05150 PAUSE T1
05155 GOTO 1330
05200 REM '4' .....
05205 X=USR(T)
05210 PAUSE T1
05215 X=USR(T)
05220 PAUSE T1
05225 X=USR(T)
05230 PAUSE T1
05235 X=USR(T)
05240 PAUSE T1
05245 X=USR(3*T)
05250 PAUSE T1
05255 GOTO 1330
05300 REM '5' .....
05305 X=USR(T)
05310 PAUSE T1
05315 X=USR(T)
05320 PAUSE T1
05325 X=USR(T)
05330 PAUSE T1
05335 X=USR(T)
05340 PAUSE T1
05345 X=USR(T)
05350 PAUSE T1
05355 GOTO 1330
05400 REM '6' .....
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
05405 X=USR(3*T)
05410 PAUSE T1
05415 X=USR(T)
05420 PAUSE T1
05425 X=USR(T)
05430 PAUSE T1
05435 X=USR(T)
05440 PAUSE T1
05445 X=USR(T)
05450 PAUSE T1
05455 GOTO 1330
05500 REM '7' ----.
05505 X=USR(3*T)
05510 PAUSE T1
05515 X=USR(3*T)
05520 PAUSE T1
05525 X=USR(T)
05530 PAUSE T1
05535 X=USR(T)
05540 PAUSE T1
05545 X=USR(T)
05550 PAUSE T1
05555 GOTO 1330
05600 REM '8' -----.
05605 X=USR(3*T)
05610 PAUSE T1
05615 X=USR(3*T)
05620 PAUSE T1
05625 X=USR(3*T)
05630 PAUSE T1
05635 X=USR(T)
05640 PAUSE T1
05645 X=USR(T)
05650 PAUSE T1
05655 GOTO 1330
05700 REM '9' -----.
05705 X=USR(3*T)
05710 PAUSE T1
05715 X=USR(3*T)
05720 PAUSE T1
05725 X=USR(3*T)
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERH < CONT'D >

```
05730 PAUSE T1
05735 X=USR(3*T)
05740 PAUSE T1
05745 X=USR(T)
05750 PAUSE T1
05755 GOTO 1330
05800 REM '/ ' .....
05805 X=USR(3*T)
05810 PAUSE T1
05815 X=USR(3*T)
05820 PAUSE T1
05825 X=USR(3*T)
05830 PAUSE T1
05835 X=USR(T)
05840 PAUSE T1
05845 X=USR(T)
05850 PAUSE T1
05855 X=USR(T)
05860 PAUSE T1
05865 GOTO 1330
05900 REM '/ ' .....
05905 X=USR(3*T)
05910 PAUSE T1
05915 X=USR(T)
05920 PAUSE T1
05925 X=USR(3*T)
05930 PAUSE T1
05935 X=USR(T)
05940 PAUSE T1
05945 X=USR(3*T)
05950 PAUSE T1
05955 X=USR(T)
05960 PAUSE T1
05965 GOTO 1330
06000 GOTO 1330
06100 GOTO 1330
06200 GOTO 1330
06300 REM '? ' .....
06305 X=USR(T)
06310 PAUSE T1
06315 X=USR(T)
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
06320 PAUSE T1
06325 X=USR(3*T)
06330 PAUSE T1
06335 X=USR(3*T)
06340 PAUSE T1
06345 X=USR(T)
06350 PAUSE T1
06355 X=USR(T)
06360 PAUSE T1
06365 GOTO 1330
06400 GOTO 1330
06500 REM 'A' ..
06505 X=USR(T)
06510 PAUSE T1
06515 X=USR(3*T)
06520 PAUSE T1
06525 GOTO 1330
06600 REM 'B' ....
06605 X=USR(3*T)
06610 PAUSE T1
06615 X=USR(T)
06620 PAUSE T1
06625 X=USR(T)
06630 PAUSE T1
06635 X=USR(T)
06640 PAUSE T1
06645 GOTO 1330
06700 REM 'C' ....
06705 X=USR(3*T)
06710 PAUSE T1
06715 X=USR(T)
06720 PAUSE T1
06725 X=USR(3*T)
06730 PAUSE T1
06735 X=USR(T)
06740 PAUSE T1
06747 GOTO 1330
06800 REM 'D' ...
06805 X=USR(3*T)
06810 PAUSE T1
06815 X=USR(T)
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
06820 PAUSE T1
06825 X=USR(T)
06830 PAUSE T1
06835 GOTO 1330
06900 REM 'E' .
06905 X=USR(T)
06910 PAUSE T1
06915 GOTO 1330
07000 REM 'F' ....
07005 X=USR(T)
07010 PAUSE T1
07015 X=USR(T)
07020 PAUSE T1
07025 X=USR(3*T)
07030 PAUSE T1
07035 X=USR(T)
07040 PAUSE T1
07045 GOTO 1330
07100 REM 'G' _..
07105 X=USR(3*T)
07110 PAUSE T1
07115 X=USR(3*T)
07120 PAUSE T1
07125 X=USR(T)
07130 PAUSE T1
07135 GOTO 1330
07200 REM 'H' ....
07205 X=USR(T)
07210 PAUSE T1
07215 X=USR(T)
07220 PAUSE T1
07225 X=USR(T)
07230 PAUSE T1
07235 X=USR(T)
07240 PAUSE T1
07245 GOTO 1330
07300 REM 'I' ..
07305 X=USR(T)
07310 PAUSE T1
07315 X=USR(T)
07320 PAUSE T1
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
07325 GOTO 1330
07400 REM 'J' .----
07405 X=USR(T)
07410 PAUSE T1
07415 X=USR(3*T)
07420 PAUSE T1
07425 X=USR(3*T)
07430 PAUSE T1
07435 X=USR(3*T)
07440 PAUSE T1
07445 GOTO 1330
07500 REM 'K' _._
07505 X=USR(3*T)
07510 PAUSE T1
07515 X=USR(T)
07520 PAUSE T1
07525 X=USR(3*T)
07530 PAUSE T1
07535 GOTO 1330
07600 REM 'L' .---
07605 X=USR(T)
07610 PAUSE T1
07615 X=USR(3*T)
07620 PAUSE T1
07625 X=USR(T)
07630 PAUSE T1
07635 X=USR(T)
07640 PAUSE T1
07645 GOTO 1330
07700 REM 'M' _
07705 X=USR(3*T)
07710 PAUSE T1
07715 X=USR(3*T)
07720 PAUSE T1
07725 GOTO 1330
07800 REM 'N' _
07805 X=USR(3*T)
07810 PAUSE T1
07815 X=USR(T)
07820 PAUSE T1
07825 GOTO 1330
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN. < CONT'D >

```
07900 REM 'O' ---
07905 X=USR(3*T)
07910 PAUSE T1
07915 X=USR(3*T)
07920 PAUSE T1
07925 X=USR(3*T)
07930 PAUSE T1
07935 GOTO 1330
08000 REM 'P' ----
08005 X=USR(T)
08010 PAUSE T1
08015 X=USR(3*T)
08020 PAUSE T1
08025 X=USR(3*T)
08030 PAUSE T1
08035 X=USR(T)
08040 PAUSE T1
08045 GOTO 1330
08100 REM 'Q' ----
08105 X=USR(3*T)
08110 PAUSE T1
08115 X=USR(3*T)
08120 PAUSE T1
08125 X=USR(T)
08130 PAUSE T1
08135 X=USR(3*T)
08140 PAUSE T1
08145 GOTO 1330
08200 REM 'R' ...
08205 X=USR(T)
08210 PAUSE T1
08215 X=USR(3*T)
08220 PAUSE T1
08225 X=USR(T)
08230 PAUSE T1
08235 GOTO 1330
08300 REM 'S' ...
08305 X=USR(T)
08310 PAUSE T1
08315 X=USR(T)
08320 PAUSE T1
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MORSCII - KERN < CONT'D >

```
08325 X=USR(T)
08330 PAUSE T1
08335 GOTO 1330
08400 REM 'T' _
08405 X=USR(3*T)
08410 PAUSE T1
08415 GOTO 1330
08500 REM 'U' ...
08505 X=USR(T)
08510 PAUSE T1
08515 X=USR(T)
08520 PAUSE T1
08525 X=USR(3*T)
08530 PAUSE T1
08535 GOTO 1330
08600 REM 'U' ....
08605 X=USR(T)
08610 PAUSE T1
08615 X=USR(T)
08620 PAUSE T1
08625 X=USR(T)
08630 PAUSE T1
08635 X=USR(3*T)
08640 PAUSE T1
08645 GOTO 1330
08700 REM 'W' ...
08705 X=USR(T)
08710 PAUSE T1
08715 X=USR(3*T)
08720 PAUSE T1
08725 X=USR(3*T)
08730 PAUSE T1
08735 GOTO 1330
08800 REM 'X' ...
08805 X=USR(3*T)
08810 PAUSE T1
08815 X=USR(T)
08820 PAUSE T1
08825 X=USR(T)
08830 PAUSE T1
08835 X=USR(3*T)
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: BASICLOCK - KERN

```
00100 REM BASICLOCK
00110 REM WRITE MACHINE LANGUAGE SUBROUTINE FROM DATAFILE INTO ME
MORY
00120 FOR A=S TO E
00130 POKE A,D(A-S)
00140 NEXT A
00150 REM WRITE ADDRESS OF SUBROUTINE INTO CLOCK INTERRUPT VECTOR
(040,040)
00160 POKE 8224,0
00170 POKE 8225,108
00180 REM CONFIGURE X-BASIC INTERPRETER TO ACCEPT CLOCK-INTERRUPT
PROCESSING
00190 POKE 8908,193
00200 POKE 8909,131
00210 POKE 8910,129
00220 POKE 13817,129
00230 REM CLOCK SHOULD BE RUNNING AT THIS POINT
00300 REM SET TIME
00310 REM
00320 REM THE NEXT FOUR MEMORY LOCATIONS -- CORRESPONDING TO THE
HOUR,
00330 REM MINUTES AND SECONDS COUNTERS, AND THE AM/PM FLAG -- WIL
L
00340 REM VARY ACCORDING TO THE ADDRESS AT WHICH HIGH MEMORY IS R
ESERVED
00350 REM FOR MACHINE LANGUAGE SUBROUTINES.
00360 REM
00370 INPUT "SET EXACT SECOND (PRESS 'RETURN' AT MARK): ";T
00380 POKE 27780,T
00390 INPUT "SET MINUTE: ";T
00400 POKE 27779,T
00410 Z=T+5:IF Z > 59 THEN Z=Z-60
00420 INPUT "SET HOUR: ";T
00430 POKE 27778,T
00440 LINE INPUT "AM OR PM? ";Q$
00450 IF Q$ < "P" THEN POKE 27781,0:GOTO 470
00460 POKE 27781,255
00470 REM DISPLAY TIME
00480 FOR I=1 TO 11
00490 PRINT
00500 NEXT I
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: BASICLOCK - KERN < CONT'D >

```
00510 H=PEEK(27778)
00520 M=PEEK(27779)
00530 C=PEEK(27780)
00540 F=PEEK(27781)
00550 PRINT TAB(20);
00560 IF H < 10 THEN PRINT " ";
00570 PRINT MID$(STR$(H),2);TAB(22);":";
00580 IF M < 10 THEN PRINT "0";
00590 PRINT MID$(STR$(M),2);TAB(24);":";
00600 IF C < 10 THEN PRINT "0";
00610 PRINT MID$(STR$(C),2);TAB(27);
00620 IF F < 255 THEN PRINT "A.M.";GOTO 640
00630 PRINT "P.M.";
00640 FOR I=1 TO 15
00650 PRINT CHR$(8);
00660 NEXT I
00670 PAUSE 200
00680 IF M <> Z THEN 510
00690 REM
00700 REM IT IS NOT NECESSARY TO RECONFIGURE THE BASIC INTERPRETE
R WITH
00710 REM THE FOLLOWING VALUES AT THE END OF THE PROGRAM RUN. IF
THESE
00720 REM BYTES ARE LEFT AS THEY WERE SET IN THE FIRST PART OF TH
IS
00730 REM PROGRAM, THE TIME-OF-DAY CLOCK WILL CONTINUE TO RUN AFT
ER
00740 REM BASICLOCK HAS ENDED. HOWEVER THE BASIC INTERPRETER M
U S T
00750 REM BE RECONFIGURED WITH THE VALUES GIVEN IN THE NEXT SIX I
NSTRUCTIONS
00760 REM BEFORE A LOAD OR DUMP TO MAG TAPE IS EXECUTED.
00770 REM
00780 POKE 8908,192
00790 POKE 8909,130
00800 POKE 8910,128
00810 POKE 13817,128
00820 POKE 8224,237
00830 POKE 8225,67
00840 END
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TEXT HANDLER - GREENHALGH

```
00010 REM TEXT HANDLER - ROGER GREENHALGH
00020 REM
00030 INPUT "NUMBER OF LINES ";P
00040 DIM I$(P),O$(P):N=1:Z=8437
00050 INPUT "LINE LENGTH ";L
00060 C$="" :C1=0:C2=0
00070 LINE INPUT "COMMAND ";C$
00080 IF C$="N" THEN PRINT TAB(L)"*":GOTO 240
00090 INPUT "FROM ";C1
00100 IF C1=0 THEN A1=1:A2=N-1:GOTO 140
00110 INPUT "TO ";C2
00120 IF C2=0 THEN A1=C1:A2=C1:GOTO 140
00130 A1=C1:A2=C2
00140 IF C$="I" GOTO 330
00150 IF C$="U" GOTO 430
00160 IF C$="J" GOTO 490
00170 IF C$="P" GOTO 590
00180 IF C$="Q" GOTO 680
00190 IF C$="E" THEN END
00200 IF C$="M" GOTO 650
00210 IF C$="C" GOTO 720
00220 IF C$="S" GOTO 370
00230 GOTO 60
00240 INPUT "GOTO ";C1
00250 IF C1<>0 GOTO 310
00260 PRINT "INPUT"
00270 LINE INPUT ;I$(N)
00280 N=N+1
00290 IF I$(N-1)="EXIT" THEN I$(N-1)="":GOTO 60
00300 GOTO 270
00310 LINE INPUT "INPUT";I$(C1)
00320 GOTO 60
00330 FOR A=A1 TO A2
00340 PRINT A;" "I$(A)
00350 NEXT A
00360 GOTO 60
00370 INPUT "INTO ";C3
00380 FOR A=A1 TO A2
00390 O$(C3+A-A1)=O$(A)
00400 O$(A)="
00410 NEXT A
```

RALEIGH N.C.

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TEXT HANDLER - GREENHALGH < CONT'D >

```
00420 GOTO 60
00430 FOR F=A1 TO A2
00440 LET S#=I$(F)
00450 GOSUB 1040
00460 LET I$(F)=S#
00470 NEXT F
00480 GOTO 330
00490 S3$=""
00500 FOR F=A1 TO A2
00510 LET S#=I$(F)
00520 GOSUB 810
00530 LET O$(F)=S#
00540 NEXT F
00550 IF S3$="" GOTO 680
00560 A2=A2+1
00570 S$="":GOSUB 810
00580 O$(A2)=S$:GOTO 550
00590 POKE Z,60:POKE Z+1,142
00600 FOR A=A1 TO A2
00610 PRINT O$(A)
00620 NEXT A
00630 POKE Z,73:POKE Z+1,32
00640 GOTO 60
00650 FOR A=A1 TO A2
00660 O$(A)=I$(A)
00670 NEXT A
00680 FOR A=A1 TO A2
00690 PRINT A;" ";O$(A)
00700 NEXT A
00710 GOTO 60
00720 FOR A=A1 TO A2
00730 O$(A)=I$(A)
00740 B=LEN(O$(A))
00750 D=INT((L-B)/2)
00760 FOR F=1 TO D
00770 O$(A)=" "+O$(A)
00780 NEXT F
00790 NEXT A
00800 GOTO 680
00810 S#=S3$+S$:S3$=""
00820 S2$="":B=LEN(S#)
```

■ ■ ■



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: INTELLIGENT DISASSEMBLER - WARNER

```
10000 REM ***** THE 'INTELLIGENT' 8080 DISASSEMBLER *****
10010 PRINT :PRINT :PRINT "THE 'INTELLIGENT' 8080 DISASSEMBLER"
10030 PRINT "    BY JAMES C. WARNER":PRINT :PRINT
10040 C$="":F$="":CNTRL 0,65000
10100 LINE INPUT "START? ";P$:GOSUB 65200:IF LEN(P$)<6 GOTO 10100
10110 S$=P$:S=P
10200 LINE INPUT "END? ";P$:GOSUB 65200:IF LEN(P$)<6 GOTO 10200
10210 E$=P$:E=P:P=S
10300 IF S>E THEN PRINT "ERR! START > END":GOTO 10100
10400 LINE INPUT "MODE? ";I$:C$=LEFT$(I$,1):IF I$="SYMBOLIC" GOTO
    30000
10500 IF I$<>"ABSOLUTE" GOTO 10400
19998 REM
19999 REM ** ABSOLUTE **
20000 LINE INPUT "FLAG? ";F$:IF LEN(F$)<>1 GOTO 20000
20100 GOSUB 65500:A1$=""
20200 Q1$="":GOSUB 60000
20300 IF LEFT$(A$,3)<>A1$ THEN A1$=LEFT$(A$,3):PRINT A1$ ".....
    ....."
20400 A$="."+MID$(A$,4):ON T+1 GOTO 20600,20600,20600,20500,20500
    ,20500,20500
20500 A$=F$+MID$(A$,2):IF T=3 THEN Q1$=Q$+"Q"
20600 ON L GOTO 20900,20800,20700
20700 Q1$=Q3$+Q2$+"A":GOTO 20900
20800 Q1$=Q2$+"Q"
20899 REM ! PRINT DISASSEMBLED LINE !
20900 PRINT A$ " " 0$ TAB(9) Q1$:P=P+L:IF P<E GOTO 20200
20990 GOTO 65534
29998 REM
29999 REM ** SYMBOLIC **
30000 LINE INPUT "VALID LO <"/> HI? ";I$:IF MID$(I$,7,1)<>"/>" GOTO
    30000
30010 IF LEN(I$)<>13 GOTO 30000
30020 S1$=LEFT$(I$,6):P$=S1$:GOSUB 65200:IF LEN(P$)<6 GOTO 30000
30030 E1$=RIGHT$(I$,6):P$=E1$:GOSUB 65200:IF LEN(P$)<6 GOTO 30000
30040 IF S1$>E1$ THEN PRINT "ERR! LO > HI":GOTO 30000
30050 N1$="SYMBOL TABLE ":N2$=" IN PROGRESS...":PRINT "PASS ONE"
    N2$:
30060 I1=VAL(S$(0))+1
30070 FOR I=0 TO I1:S$(I)="":NEXT I
30100 W0=2:W1=4:W2=2
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: INTELLIGENT DISASSEMBLER - WARNER < CONT'D >

```
30998 REM
30999 REM ** SYMBOL TABLE CREATION **
31000 S$(0)=STR$(2):T$="1":S$(1)=CHR$(1)+S$+T$:S$(2)=CHR$(1)+E$+T$
P=5
31100 GOSUB 50000:P=P+L:IF P<E GOTO 31100
31998 REM
31999 REM ** SYMBOL TABLE EVALUATION **
32000 PRINT :PRINT N1$ "EVALUATION" N2$:
32099 REM ! POINT TO START !
32100 FOR I=1 TO VAL(S$(0)):IF MID$(S$(I),2,6)<S$ THEN NEXT I
32199 REM ! PRIMARY EVALUATION LOOP !
32200 FOR I=I TO VAL(S$(0)):IF MID$(S$(I),2,6)<E$ THEN GOSUB 49000
0:IF T(1)<>0 THEN NEXT I
32300 IF MID$(S$(I),2,6)>=E$ GOTO 33000
32399 REM ! GOT NON-BRANCH ENTRY !
32400 I2=I:P$=MID$(S$(I2),2,6):GOSUB 65300:S1=P
32500 I3=I2+1:P$=MID$(S$(I3),2,6):GOSUB 65300:E1=P:T1=0
32699 REM ! GOT POTENTIAL 'DATA' RANGE !
32700 P=S1:GOSUB 57000:S1=S1+L:IF S1<E1 GOTO 32700
32710 IF T1>0 GOTO 32900
32720 IF T1=0 THEN S$(I2)=CHR$(2)+MID$(S$(I2),2):GOTO 32900
32789 REM ! GOT CONFIRMED 'DATA' AREA !
32790 S$(I2)=CHR$(3)+MID$(S$(I2),2):P$=MID$(S$(I2),2,6):GOSUB 65300:S1=P:L=0
32799 REM ! RE-RE-DISASSEMBLE 'DATA' AREA !
32800 P=P+L:T1=0:GOSUB 57000:L1=L:IF A$<E$ THEN ON L GOTO 32800,32830,32810
32810 IF T1>0 THEN IF A$<>MID$(S$(I2),2,6) THEN T$="3":GOSUB 55000:GOTO 32200
32819 REM ! DISASSEMBLE BYTES 2-3 IF MULTI-BYTE INSTR'S !
32820 P=P+1:T1=0:GOSUB 57000:IF T1>0 THEN T$="3":GOSUB 55000:GOSUB 50000:I=I2+1:GOTO 32200
32830 P=P+1:T1=0:GOSUB 57000:IF T1>0 THEN T$="3":GOSUB 55000:GOSUB 50000:I=I2+1:GOTO 32200
32890 S1=S1+L1:IF S1<E1 THEN P=S1:L=0:GOTO 32800
32899 REM ! CHK FOR ADD'L ENTRIES !
32900 IF I3<VAL(S$(0)) THEN IF MID$(S$(I3),2,6)<E$ THEN I=I3:GOTO 32200
32998 REM
32999 REM ** SYMBOL TABLE LABEL ASSIGNMENT **
33000 PRINT :PRINT N1$ "LABEL ASSIGNMENT" N2$:
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: INTELLIGENT DISASSEMBLER - WARNER < CONT'D >

```
33099 REM ! ENTRIES < START = 'EXT.' !
33100 FOR I=1 TO VAL(S$(0)):IF MID$(S$(I),2,6)<S$ THEN GOSUB 4800
0:S$(I)=LEFT$(S$(I),7)+"EXT."+I$:NEXT I
33199 REM ! ENTRIES TWIX START & END = TBL FLG !
33200 GOSUB 48000
33210 IF LEFT$(S$(I),1)=CHR$(1) THEN S$(I)=LEFT$(S$(I),7)+"RTN."+
I$:GOTO 33290
33220 IF LEFT$(S$(I),1)=CHR$(2) THEN S$(I)=LEFT$(S$(I),7)+"UNK."+
I$:GOTO 33290
33230 S$(I)=LEFT$(S$(I),7)+"DAT."+I$
33290 I=I+1:IF MID$(S$(I),2,6)<E$ GOTO 33200
33299 REM ! ENTRIES > END = 'EXT.' !
33300 FOR I=1 TO VAL(S$(0)):GOSUB 48000:S$(I)=LEFT$(S$(I),7)+"EXT
."+I$:NEXT I
33998 REM
33999 REM ** FINAL SYMBOLIC PASS **
34000 F$="X":PRINT :PRINT "FINAL PASS BEGINNING. ":GOSUB 65500:F
$=""
34099 REM ! PRINT EXT LBLs UP TO START !
34100 FOR I1=1 TO VAL(S$(0)):IF MID$(S$(I1),2,6)<S$ THEN GOSUB 47
000:NEXT I1
34199 REM ! SET ORGIN !
34200 PRINT LEFT$(S$,3) ". " RIGHT$(S$,3) TAB(30) "ORG " S$ "A"
34300 L9$=MID$(S$(I1),8):P=5
34999 REM ** INSTRUCTION DECODE **
35000 L$="":L1$="":Q1$="":GOSUB 60000
35100 IF A$>MID$(S$(I1),2,6) THEN GOSUB 43000:GOTO 35100
35200 IF A$<>MID$(S$(I1),2,6) GOTO 35500
35300 IF LEFT$(S$(I1),1)=CHR$(3) GOTO 36100
35400 L$=MID$(S$(I1),8):I1=I1+1
35500 IF T=3 THEN L1$=Q$+"Q"+N$
35600 ON L GOTO 35700,35620,35610
35610 GOSUB 41000:GOTO 35700
35620 Q1$=Q2$+"Q"
35699 REM ! PRINT DISASSEMBLED LINE !
35700 GOSUB 40000:PRINT Q2$ " " Q3$ TAB(22) L$ TAB(30) Q$ TAB(35)
L1$ Q1$
35710 P=P+L:IF P<E GOTO 35000
35720 GOSUB 60000
35730 IF A$>MID$(S$(I1),2,6) THEN GOSUB 43000:IF I1<VAL(S$(0)) GO
TO 35730
35799 REM ! PRINT REMAINING EXT LBLs !
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: INTELLIGENT DISASSEMBLER - WARNER < CONT'D >

```
35800 IF I1<VAL(S$(0)) THEN GOSUB 47000:I1=I1+1:GOTO 35800
35899 REM ! PRINT END STMT !
35900 A=P:GOSUB 60500:PRINT LEFT$(A$,3) "." RIGHT$(A$,3) TAB(30)
"END " L9$
35990 GOTO 65534
36000 REM ** DATA DECODE **
36100 GOSUB 40000:PRINT TAB(22);
36200 IF A$=MID$(S$(I1),2,6) THEN PRINT MID$(S$(I1),8);:I1=I1+1
36300 PRINT TAB(30) "DB " Q$ "Q":P=P+1:GOSUB 60000
36500 IF A$<MID$(S$(I1),2,6) GOTO 36100
36600 IF A$=MID$(S$(I1),2,6) THEN IF LEFT$(S$(I1),1)=CHR$(3) GOTO
36100
36900 GOTO 35000
39997 REM
39998 REM *** SYMBOLIC SUPPORT ***
39999 REM ** LINE PREFIXER **
40000 PRINT LEFT$(A$,3) "." RIGHT$(A$,3) " " Q$ " ";:RETURN
40998 REM
40999 REM ** LABEL RETRIEVER **
41000 FOR I=1 TO VAL(S$(0)):IF Q3$+Q2$(<)MID$(S$(I),2,6) THEN NEXT
I
41100 IF I>VAL(S$(0)) THEN L1$=Q3$+Q2$+"A"+N$:RETURN
41200 L1$=MID$(S$(I),8):RETURN
42998 REM
42999 REM ** CATCH-UP EQUATER **
43000 A=P:P$=MID$(S$(I1),2,6):GOSUB 65300:I=P-A:GOSUB 48000:P=A
43100 PRINT MID$(S$(I1),2,3) "." MID$(S$(I1),5,3) TAB(22) MID$(S$(
I1),8) TAB(30)"EQU *" I$:I1=I1+1:RETURN

46998 REM
46999 REM ** EQUATER **
47000 PRINT MID$(S$(I1),2,3) "." MID$(S$(I1),5,3) TAB(22) MID$(S$(
I1),8) TAB(30)"EQU " MID$(S$(I1),2,6) "
A": RETURN
47998 REM
47999 REM ** LABEL SUFFIXER **
48000 I$=STR$(I):I$=LEFT$(I$,LEN(I$)-1):I$=RIGHT$(I$,LEN(I$)-1):R
ETURN
48998 REM
48999 REM ** TABLE IMPACT IDENTIFIER **
49000 FOR I1=0 TO 4:T(I1)=0:NEXT I1
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: INTELLIGENT DISASSEMBLER - WARNER < CONT'D >

```
49100 FOR I1=8 TO LEN(MID$(S$(I),8))+7:T(VAL(MID$(S$(I),I1,1)))=T
(VAL(MID$(S$(I),I1,1)))+1:NEXT I1
49200 RETURN
49998 REM
49999 REM ** SYMBOL TABLE BUILD MAYBE **
50000 GOSUB 60000:ON T+1 GOTO 55900,50700,50700,55900,50600,50500
,55900
50500 T$="1":A$=Q3$+Q2$:GOSUB 55000:T$="4"
50600 A=P+L:GOSUB 60500:GOTO 55000
50700 A$=Q3$+Q2$
50900 REM
54999 REM ** SYMBOL TABLE UPDATE **
55000 FOR I=1 TO VAL(S$(0)):IF MID$(S$(I),2,6)<A$ THEN NEXT I
55099 REM ! UPDATE EXISTING !
55100 IF A$=MID$(S$(I),2,6) THEN S$(I)=S$(I)+T$:GOTO 55900
55200 IF A$<MID$(S$(I),2,6) GOTO 55400
55299 REM ! ADD NEW LAST !
55300 S$(I)=CHR$(1)+A$+T$:S$(0)=STR$(I):GOTO 55900
55399 REM ! SHUFFLE DOWN !
55400 FOR I1=VAL(S$(0))+1 TO I STEP -1:S$(I1)=S$(I1-1):NEXT I1
55499 REM ! INSERT NEW !
55500 S$(I)=CHR$(1)+A$+T$:S$(0)=STR$(VAL(S$(0))+1)
55900 RETURN
56998 REM
56999 REM ** INSTRUCTION EVALUATOR **
57000 GOSUB 60000:ON T+1 GOTO 57900,57100,57200,57300,57900,57100
,57400
57100 IF Q3$+Q2$>=S1$ THEN IF Q3$+Q2$<=E1$ THEN T1=T1+W1:GOTO 579
00
57110 T1=T1-(W1/W0):GOTO 57900
57200 IF Q3$+Q2$>=S1$ THEN IF Q3$+Q2$<=E1$ THEN T1=T1+W2:GOTO 579
00
57210 T1=T1-(W2/W0):GOTO 57900
57300 T1=T1-(W1*W0):GOTO 57900
57400 T1=T1-(W2*W0)
57900 RETURN
59997 REM
59998 REM *** UNIVERSAL SUPPORT ***
59999 REM ** FULL DECODE **
60000 I=PEEK(P):A=P:Q2$="":Q3$=""
60100 Q$=LEFT$(T$(I),3):L=VAL(MID$(T$(I),4,1)):T$=MID$(T$(I),5,1)
:T=VAL(T$):Q$=MID$(T$(I),6):ON L GOTO 605
```





# THE 'INTELLIGENT' DISASSEMBLER

By: Jim Warner  
10121 Hyway 55  
Plymouth MA 55441

OK, so maybe we don't need another 8080 disassembler. It's true that the second issue of REMark contained a disassembler program which could serve us very well. But frankly, I've been intrigued with the concept of a disassembler almost since the day I first keyed in and successfully ran the Heath initial test routine.

I remember that event very clearly. It took place at 4:00 a.m. after working 'round the clock' completing the front panel board. What a thrill to realize for the first time that here was my very own computer — and it was actually doing something! My excitement increased as each new Heath distribution tape was loaded and run. Now my computer was not only doing something, but it was doing something useful. Of course, that realization brought a flood of questions, mostly of the "wow, how does it do that?" variety.

Not being one to leave such questions unanswered, I set about to create a disassembler, in assembly language no less. Believe me, there's no better way to learn a new computer language than to tackle such a project. After several weeks, I had my first version up and "sometimes" running. It was a modest beginning. This disassembler was only capable of absolute disassembly. There were no provisions for symbolic labels and I frequently encountered the situation where data was being misinterpreted as instructions. Wouldn't it be nice, I thought, if the disassembler could make that distinction. I mean if I was able to apply standards of reasonableness to what I watched being disassembled, why couldn't he? That way he'd know that at a particular memory location was an ASCII "1", not a load stack pointer immediate instruction.

The 8080 instruction set contains a mere 245 separate instructions. Thus, of the 256 different bit configurations possible in an 8-bit byte, only 11 combinations constitute a non-instruction. And even those non-instructions could be valid as the second or third byte of a multi-byte instruction. So, we have a situation where virtually any random memory location would generate what looked like a valid 8080 instruction. Yet we know that a good portion of a computer program is really data — tables and literals.

At last, here was a project worthy of my attention — a clearly impossible task! I would create an "intelligent" disassembler, one which was capable of distinguishing between object program logic (instructions) and object program data. It would assign default labels to program and data segments plus provide for user designated labels as well. Its output would be acceptable input to the text editor (TED-8) for potential modification as well as the assembler (HASL-8) for re-assembly.

After investing many weeks in pursuit of this glorious hunk of software, it began to appear as though the task was indeed impossible. Version after longer, more patched version crashed and proved very difficult to debug. The need for frequent revisions along with the subsequent lengthy assemblies clearly suggested flaws in my original design. And these flaws prevented me from ever actually testing my key algorithms — those dealing with symbol table manipulation and access. My dream was slipping away! What I desperately needed was a means to make program modifications quickly and painlessly. I also needed a powerful run-time package which could monitor or alter program activity on the fly. Hey, wait a minute. What I needed was the Extended Benton Harbor BASIC interpreter!

Though I remain the type of person who likes the informal title of "assembly language programmer", my BASIC interpreter will never again be thought of in terms of last resort. Rather, it will be thought of as perhaps the best means of prototyping any complex task. I, therefore, offer this BASIC program as a permanent reminder to me, and a suggestion to other readers, of the power and flexibility of this sometimes snubbed high level language. And, I might add as an example of the latest Extended BASIC with data base capabilities.

So, before I get back to HASL-8 with this BASIC prototype to use as a model in developing my faster, more efficient assembly language version, let me share with you its inner workings. I think you'll find that this is not just another disassembler.

## GENERAL BACKGROUND

The "intelligent" disassembler is designed to run under Extended Benton Harbor BASIC version 10.02.01. It consists of program text and a data base. The program text occupies approximately 6500 bytes of memory, but can be reduced to less than 5000 by deleting the REMark statements, all of which are dispensable without additional modification. The data base will occupy additional memory, but its size is variable — more on that later.

After an FLOAD, the CONTINUE command will result in an identification heading followed by a "START?" prompt. This invites the disassembly start address. All disassembler addresses are in offset octal format and require a full 6 digits, including leading zeros. The same is true of the "END?" prompt which follows a valid start designation.

After the disassembly range has been established, the "MODE?" prompt is printed. There are two distinct modes of operation under this disassembler ("SYMBOLIC" and "ABSOLUTE") and these are the only valid responses.

## ABSOLUTE MODE

When this mode is specified, the next prompt is "FLAG?". Any single character input at this point will be used to alert you when disassembly has or might become suspect — when instructions are encountered which may signal a data area instead. For a description of such instruction types (in this case types #3-6), see Table 1.

The final prompt will be an invitation to configure your terminal and type Return. In the absolute mode the disassembler formats the print line so as to take advantage of Heath's H9 Short Form provision. In this way a larger area of memory can be displayed at any given time.

Each disassembled line is preceded by the octal representation of the low order byte of the current memory location. In some cases this prefix will also display your designated "flag." When page boundaries change (the high order byte of the current memory location) an additional line will be generated to so indicate.

At any point during the disassembly range, a CTL-B will result in a termination of the current process and the "P=" prompt will be printed. This is an invitation to alter the disassembler's current memory pointer (again in offset octal format) so as to advance or back up the disassembly process. This can be a convenient way to investigate a subroutine following a "CALL" instruction. When you wish to resume the abandoned memory location, another CTL-B followed by the new address will accomplish it.

## SYMBOLIC MODE

When this mode is specified, the next prompt following the mode designation is "VALID LO '/' HI?". We'll discuss the reason for this later. For now suffice it to say that this requires two offset octal addresses separated by a slash (/). Depending on the length of your disassembly, the next prompt could be some time is arriving. However, here's what you will see at the terminal, at varying times of course.

```
PASS ONE IN PROGRESS...
```

```
SYMBOL TABLE EVALUATION IN PROGRESS...
```

```
SYMBOL TABLE LABEL ASSIGNMENT IN PROGRESS...
```

```
FINAL PASS BEGINNING. CONFIGURE TERMINAL.  
TYPE CR:"
```

Since we will talk about each phase in detail, at this point let me just briefly cover the symbolic CTL-B options.

By typing CTL-B just prior to the final pass, the disassembler will print the symbol table, including addresses and labels, for you to review. A second CTL-B while such a review is in progress will terminate the printout and return you to the configure terminal prompt. This process can be repeated unendingly if you desire.

## DATA BASE (DRIVING TABLES)

The intelligent disassembler data base consists of several items — minus the symbol table. It is up to each user to dimension the symbol table to whatever size memory will support. This is accomplished by using the "DIM S\$(xxx)" in the command mode. Care must be taken to leave room for the growth of the symbol table entries also. Thereafter, assuming the program is saved at this point (FDUMP), the data base would consist of T\$(255), T(4), N\$, S\$(xxx).

The string variable N\$ is simply an error note. Its value can easily be changed in the command mode. To find out how it's used, you might try a "SYMBOLIC" disassembly with a start of 000256 and an end of 000264. For this experiment you may specify any "VALID LO '/' HI" addresses.

The small numeric array T is used only during symbol table evaluation. Its elements are zero at this point.

The large string array T\$ represents the primary disassembler table. Unlike the other members of the data base, it is crucial to both disassembly modes.

This table enables the disassembler to produce octal values, determine instruction lengths and types and print the mnemonic operation codes. It consists of variable length entries which contain the following information:

LEFT\$(T\$(x),3) = Octal equivalent of x

MID\$(T\$(x),4,1) = 8080 instruction length

MID\$(T\$(x),5,1) = Disassembler instruction type

MID\$(T\$(x),6) = 8080 operation code

Thus the entry for T\$(205) would appear literally as "31531CALL" while the entry for T\$(1) would look like "00132LXI B."

This should be fairly self-explanatory and its need vary apparent. For a more thorough review of the primary table, you might wish to issue the following immediate command after you FLOAD:

```
FOR I=0 TO 255: PRINT LEFT$(T$(I),3) " " MID$(T$(I),4,1)
```

```
" " MID$(T$(I),5,1) " " MID$(T$(I),6): NEXT I
```

Hit the Short Form button then Erase before you key Return, use CTL-S & Q to govern scrolling and review the primary table 'til your hearts content.



## THEORY OF SYMBOLIC OPERATION

We know that the CPU will execute instructions sequentially within a given block of memory until told to do otherwise. One way to accomplish that "otherwise" is through the use of a branch type instruction. Examples of assembly language branches would be "CALL (address)" and "JZ (address)". The equivalent BASIC statements would read "GOSUB (line #)" and "IF Z=0 GOTO (line #)".

There is one other type of instruction which affects the sequential nature of program execution. This is the instruction through which processing cannot pass. JMP, RET and PCHL fall into this category and constitute the only such instructions in the 8080 instruction set. As far as the CPU is concerned, the only way processing could proceed beyond these points is if the address was reached through a branch instruction found elsewhere in the object program.

A disassembler which could assign symbolic labels to the addresses generated by these two types of instructions would have taken a big step toward "intelligent" disassembly.

### PASS ONE (SYMBOL TABLE CREATION)

As we sequentially disassemble an area of memory, let's ask our disassembler to construct a table. In its inception it will be primarily an address table, one whose entries consist of addresses. However, it will eventually become our symbol table and, since that name sounds so much more impressive (and will later be more accurate), you'll hear me refer to it more often as such.

This table will contain variable length entries of the format:

LEFT\$(S\$(x),1) = Evaluation flag

MID\$(S\$(x),2,6) = Address

MID\$(S\$(x),8) = Type references/Lable

We'll insert the address of all branch objects (instruction type #1) into this table. While we're at it, let's also insert the address of all load objects (instruction type #2) into our table. Although such instructions do not affect the sequential nature of program execution, they may give a hint of a potential data pocket. Finally, we will take the address following those instructions through which processing cannot pass (current location + instruction length), these are instruction type #4, and place it in our table as well. These deferred impact instructions provide an even stronger hint of potential data pockets within the disassembled program.

Incidentally, the job of building this symbol table will be made easier if the entries are always kept in ascending order with no duplication. Such is the case with my BASIC disassembler. For the technique used, I direct your attention to lines 55000-55900 of the BASIC listing. You'll notice that element zero, S\$(0), is used to hold the total number of entries currently in the table. It will be incremented whenever a new address is added, whether that address is inserted within the table or appended to the end of the table (depending on its value). Existing entries are simply updated with the new reference type. In this way we can main-

tain a history of how many references were made to a particular entry and, more importantly, in what ways that address was used in the disassembled program.

All right, we've completed the first pass. A block of memory has been sequentially disassembled and assumed to have consisted entirely of instructions — no data. In the process we created a table of addresses. Have we done enough? Well, we've done enough if all we want is the ability to assign symbolic labels to the disassembled listing. Unfortunately, our disassembler is still "dumb" and would merrily treat data as instructions. We must now ask him to analyze those table entries and, where appropriate, re-disassemble certain object program segments. In fact, we'll find that some areas of a program must be re-disassembled (disassembled for a third time) if this guy is to acquire real intelligence.

### SYMBOL TABLE EVALUATION

In the evaluation phase, we can ignore all addresses in our symbol table which are less than the disassembly start address or greater than the disassembly end address. Such entries will be considered "external" and we won't try to classify them as pointing to data or instructions. Next, we can ignore all branch object addresses. We'll assume that the author of whatever it is we are disassembling wouldn't have directed the CPU to execute a table or literal pool (in other words data). Now you can see why it's important that each entry in the symbol table also provide an indication of how that address was used in the disassembled program.

Having ignored what is probably the majority of entries in our symbol table, what do we have left? We are left to consider only those addresses which were used in a load instruction (type #2) and those addresses generated by the disassembler because processing could not pass through the previous instruction (types #4 & #5). Remember also that these remain entries will fall within the range of our disassembly, having ignored external references.

At this point, any further disassembly will be controlled by the addresses contained in the symbol table entries we have just identified. For the purpose of re-disassembly we will develop a sub-range which begins at the address contained in one symbol table entry and extends to the address contained in the next symbol table entry. We have already disassembled such a "suspect" program segment once. What we need now is a means to weigh what we'll ask our disassembler to take a second look at — re-disassemble. This is where the full compliment of instruction "types" become important. We will also provide two additional tools to use during this phase.

Our disassembler knows the start and end points of the requested disassembly. But what if that range represented only a portion of the complete program? A branch instruction with an object address outside this range would appear less valid than one whose object fell within this range. In the later, we would be inclined to consider that an instruction whereas in the former, we might see it as data. In an effort to enhance the accuracy of such evaluations, we will give the disassembler a "valid" range, which may or may not correspond to the disassembly range (the "VALID LO '/' HI?" prompt). Now the disassembler can apply this "valid" range to his evaluation of three-byte instructions (types #1, 2 = 5).



The final tool we will provide is the means to consider any suspect program segment in its entirety. We want to avoid classifying any program area as data or instructions on the basis of a single instruction. Any such decision should be cumulative. However, since each instruction must be considered individually we will offer an evaluation variable which can be dynamically altered depending upon the instruction type and the instruction's content. This evaluation variable will be subjected to factors which combine the effect of instruction type and content.

The factors, or weights, I've chosen can be found on line 30100 of the BASIC listing. They are actually applied at lines **5700-57900**. These factors are somewhat arbitrary and you readers certainly have my permission to alter them as you see fit — we call that “tuning”. However, they seem to do the job for me. Here's how they work.

If a branch instruction (type #1) is encountered in this re-disassembly and, if the object address falls within the “valid” range, the disassembler will add a positive bias to the evaluation variable. If the object address is outside the “valid” range, a somewhat weaker negative bias is applied to the evaluation variable. The same approach is used with load instructions (type #2) but the effect on the evaluation variable is not as great. If a suspect instruction (type #6) is found, a strong negative bias is applied to the evaluation variable. Finally, if a non-instruction (remember, we're still disassembling sequentially at a location dependent upon the previous instruction's length) is encountered, type #3, the strongest negative bias is applied to the evaluation variable.

Once we complete this re-disassembly, if we find that the evaluation variable yields a positive result we can flag that symbol table entry as pointing to program logic. In reality, all symbol table entries are so flagged upon entry — innocent until proven guilty, as it were. If our evaluation variable remained zero, it means we don't have enough information to make a definite determination. We'll flag that entry as “unknown” and later, through convention, treat the program segment as instructions. However, if the evaluation variable is negative (a likely result if a non-instruction was encountered, a probable result if any “bad” branch or load instructions were found), we'll flag that symbol table entry as data. We should also ask our disassembler to find his second wind because there's more work to do with any such program segment (remember re-re-disassembly?).

There are a couple of reasons for taking what amounts to a third look at any “confirmed” data area. First, a single byte of data may have been interpreted as a multi-byte instruction during pass one. In such a case, we may run the risk of losing the beginning of an otherwise valid instruction through the interpretation of data bytes 2-3 as an address or value. This very situation was clearly illustrated in the earlier REMark disassembler at address 000033.

The second reason for re-examining this data area lies in the way our symbol table was constructed. The next entry in the table (the entry containing the ending address of this sub-range) got there for a reason. If that reason was because of JMP instruction (type #5) and, if that instruction now generates a “valid” object address, then we owe our symbol table one more entry — one corresponding to the beginning of that instruction. Such an approach allows us to properly evaluate and label the JMP instruction we find at address 000050 following the data at

address 000043. Our only risk in “forcing” entries in this way is a few extraneous labels during the final pass, with no harm done. I prefer that to too few labels.

With the evaluation phase we find the last maintenance being applied to our symbol table. We won't be adding any more entries. However, this entire process will be repeated with each non-branch table entry throughout the range of disassembly.

## SYMBOL TABLE LABEL ASSIGNMENT

The time has come (finally?) to transform our lowly address table into a true symbol table. The entries are now flagged as program logic, unknown and data. These flags will control the disassembly process (how memory is looked at) during the final pass. But what about us, the user, and the effect symbolic label content has on readability? Wouldn't it be nice if the labels we develop also reflected the distinction of program logic, etc.? Well, why not?

Let's call program logic entries “RTN” (for routine), un known segments “UNK” and data areas “DAT”. We'll call all external labels “EXT” and we will append to each label an indication of where it resides within the table. Thus, our first few entries might appear as “EXT.1”, “RTN.2”, “UNK.3” and “DAT.4”. In this way, using the CTL-B and CTL-C provisions, we could return to command mode during symbol table review and issue the following immediate command:

```
SS(2)=LEFT$(SS(2),7)+”ANYVAL”
```

Now we have replaced “RTN.2” with “ANYVAL” while retaining the evaluation flag and address. Thereafter, the disassembler would associate “ANYVAL” with value in the symbol table whenever it was used during the final pass. Thus we have a simple, effective means for user designated labels.

This technique can be used over a period of time, as program segment functions are identified by you, to make an extremely readable disassembly listing. If at any time you wished to terminate a final pass listing, and resume at the start of the final pass later, the following immediate mode commands would accomplish it: —

```
UNLOCK
```

```
1 CNTRL 0,65000: GOTO 34000
```

```
FDUMP “any name”
```

## THE FINAL PASS

With labels assigned we can now begin our final pass, producing a symbolic assembly listing for our disassembly. Actually, this pass is not unlike the very first pass except we will be printing what we find as we PEEK( ) at various memory locations. Additionally, we'll use labels instead of octal values for three- byte instruction operands and we'll sometimes preface a line with a label — when that label's address matches our current memory location. Of course, if at that point the symbol table indicates we've reached a data area. We will force the “DB” (define byte) interpretation on what we find.

There you have it, symbolic (intelligent) disassembly. Simple, wasn't it?

## PROGRAM VARIABLES

	utility address, generally = P		
A\$	utility address in offset octal, generally = P	P\$	utility offset octal address, generally user input
C\$	mode identifier control byte/CTL-B enable	Q\$	utility octal value, from T\$(I)
E	disassembly end	Q1\$	symbolic two-byte instruction operand
E\$	disassembly end, offset octal	Q2\$	multi-byte instr's byte 2 octal value, from T\$(I)
F\$	absolute flag/CTL-B enable	Q3\$	multi- Q3\$ multi-byte instr's byte 3 octal value, from T\$(I)
I,I1	utility array reference indices	S	disassembly start
I2,I3	symbolic re-disassembly symbol table reference indices	S\$	disassembly start, offset octal
I\$	utility string	S\$(I)	symbol table
L	instruction length, from T\$(I)	T,T\$	instruction type, from T\$(I)
L\$	symbolic line prefix label	T1	symbolic evaluation variable
L1\$	symbolic instruction operand label	T(I)	symbolic symbol table type?references de-code array
L9\$	symbolic end statement start label	T\$(I)	primary disassembler driving table
N\$	symbolic integrity violation message	WO-W2	symbolic evaluation weights
O\$	instruction mnemonic op code, from T\$(I)	S1,S1\$ E1,E1\$	symbolic sub-range re-disassembly start/end addresses
P	current memory pointer		





<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY GRAPH - FULLER

```
00010 REM *****
*
00020 REM *          UTILITY GRAPH
*
00030 REM *          PROGRAMMER:  WILLIAM E. FULLER
*
00040 REM *          WRITTEN IN BH. BASIC VERSION 01.02.01
*
00050 REM *          DATE:  28/OCT/78
*
00060 REM *****
*
00065 PRINT :PRINT
00070 LINE INPUT "DO YOU WISH INSTRUCTIONS (Y/N)?";I$
00075 PRINT :PRINT
00080 IF I$="Y" THEN GOSUB 40000
00100 LINE INPUT "HEADING FOR COLUMN ONE (LIMIT 6 LETTERS) ?";E1$
00110 PRINT :PRINT
00120 LINE INPUT "HEADING FOR COLUMN TWO (LIMIT 7 LETTERS) ?";E2$
00130 PRINT :PRINT
00140 LINE INPUT "HEADING FOR COLUMN THREE (LIMIT 6 LETTERS) ?";E
3$
00150 PRINT :PRINT
00160 INPUT "BEGINNING DATE (INDUSTRIAL TYPE SUCH AS 8270 ECT.) ?
";D1
00170 PRINT :PRINT
00175 INPUT "INCREMENT DATE UNITS BY ?";J
00178 PRINT :PRINT
00180 LINE INPUT "SCALE FACTOR ?";S1$
00190 PRINT :PRINT
00200 LINE INPUT "TIME UNITS IDENTIFICATION (DAYS WEEKS ECT.) ?";
T1$
00210 PRINT :PRINT
00220 LINE INPUT "TITLE FOR GRAPH (LIMIT TO 25 CHARACTERS) ?";T2$
00230 PRINT :PRINT
00240 LINE INPUT "SPECIAL NOTE ?";T3$
00250 PRINT :PRINT
00300 PRINT TAB(53)"*****"
00310 PRINT TAB(53)"*   ";T2$;TAB(82)"*"
00320 PRINT TAB(53)"*****"
00340 PRINT "NOTE: ";T3$
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY GRAPH - FULLER < CONT'D >

```

00350 PRINT "UNITS PER LINE: ";T1$
00360 PRINT "SCALE FACTOR: ";S1$
00370 PRINT "BEGINNING DATE/UNIT OF TIME: ";D1
00380 PRINT "PROGRAM BY W. E. FULLER"
00390 PRINT :PRINT
00500 GOSUB 4000
00600 GOSUB 5000
00650 READ A,B
00655 A=INT(A)
00657 B=INT(B)
00670 IF A=9999 GOTO 60000
00680 IF B=9999 GOTO 60000
00700 IF A>B THEN C=A:D=B:A=D:B=C:GOTO 1950
00720 IF A=B GOTO 2000
00800 REM *****
00810 REM THE PRINT ROUTINE FOR THE GRAPH
00820 REM *****
01900 PRINT "*"D1TAB(7)!"A;TAB(16)!"B;TAB(24)!"TAB(A+24)"0";TA
B(B+24)"X";TAB(124)*"
01920 GOSUB 6000
01925 GOTO 650
01950 PRINT "*"D1TAB(7)!"B;TAB(16)!"A;TAB(24)!"TAB(A+24)"X";TA
B(B+24)"0";TAB(124)*"
01960 GOSUB 6000
01965 GOTO 650
02000 PRINT "*"D1TAB(7)!"A;TAB(16)!"B;TAB(24)!"TAB(A+24)"Z";TA
B(124)*"
02010 GOSUB 6000
02015 GOTO 650
04000 PRINT "!"E1$TAB(7)"/";E2$TAB(16)"/";E3$;
04100 PRINT TAB(24)"0      1 0      2 0      3 0      4 0
      5 0      6 0";
04110 PRINT "      7 0      8 0      9 0      100"
04200 RETURN
05000 PRINT "=====+=====+=====+=====
==+=====+=====+=====+";
05100 PRINT "=====+=====+=====+=====+"
05200 RETURN
06000 PRINT "*=====!=====!=====!-----!-----!-----
---!-----!-----!-----!";
06100 PRINT "-----!-----!-----!-----*"

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY GRAPH - FULLER < CONT'D >

```
06150 D1=D1+J
06200 RETURN
09000 REM *****
09100 REM THE DATA FOR THE GRAPH IS PLACED INTO DATA STATEMENTS
09200 REM IN LINES 10000 THROUGH 39999
09300 REM A VALUE OF 9999 FOR A DATA VALUE WILL CAUSE PROGRAM
09400 REM TO END.
09500 REM *****
10000 DATA 10,20,20,10,20,20
10010 DATA 16,19,23,29,39,39,41,51,91,5
10020 DATA 85,10,65,30,53,45,63,58,42,66,23,79,43,92
10030 DATA 66,65,72,80,80,80,79,55,76,33,55,89,87,20
10040 DATA 65,91,8,15,19,40,17,17,25,64,18,88,72,90
10050 DATA 39,35,12,5,43,89,28,66,14,54,8,6,27,27,3,55
10060 DATA 30,20,57,5,89,20,9999,0
40000 PRINT "THIS IS A GENERAL PURPOSE GRAPHING PROGRAM. THIS PR
OGRAM"
40010 PRINT "REQUIRES A PRINTER WITH A CONSOLE LENGTH OF 132 BUT
MAY"
40020 PRINT "EASILY BE MODIFIED FOR OTHER CONSOLE WIDTHS. THE GR
APH WILL"
40030 PRINT "PLOT TWO QUANTITIES AGAINST EACH OTHER. THE FIRST C
OLUMN"
40040 PRINT "IS THE DATE OR OTHER UNIT OF TIME WHICH WILL INCREME
NT BY"
40050 PRINT "AN AMMOUNT WHICH YOU SPECIFY IN AN INPUT STATEMENT.
THE 2ND COLUMN"
40060 PRINT "IS THE 'O' VALUE WHICH IS PLOTTED ON THAT LINE. THE
THIRD"
40070 PRINT "COLUMN REPRESENTS THE 'X' VALUE WHICH IS PLOTTED ON
THE"
40080 PRINT "SAME LINE. A SINGLE 'Z' IN A LINE INDICATES THAT TH
E THE"
40090 PRINT "'X' AND 'O' VALUES ARE IDENTICAL."
40095 PRINT "ENTER DATA VALUES TO BE PLOTTED IN LINES 10000 TO 40
000."
40097 PRINT "HIT SPACE BAR WHEN YOU WISH TO CONTINUE.":PAUSE
40098 PRINT :PRINT
40100 RETURN
50000 REM *****LIST OF VARIABLES USED*****
50010 REM I$=ANS. Y/N FOR INSTRUCTIONS
```





<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE

```
00010 PRINT "THIS PROGRAM WILL COMPUTE MISCELLANEOUS ";
00015 PRINT "ELECTRONIC FORMULAS":PRINT
00018 PRINT "IF NOT SPECIFIED, ALL ENTRIES ARE IN UNITS":PRINT
00020 REM PROGRAM BY PAUL CRABTREE
00060 PRINT TAB(21)"ENTER NUMBER OF ITEM":PRINT
00090 DIM R1(20),C1(20),L1(20)
00100 FOR X=1TO 12:READ B$:IF B$="777"GOTO 105
00102 NEXT X
00105 FOR A=1TO 20:READ B$
00110 IF B$="999"GOTO 505
00115 ON A GOTO 120,125,120,125,120,125,120,125,120,125
00118 ON A-10GOTO 120,125,120,125,120,125,120,125
00120 PRINT TAB(5)B$,:GOTO 135
00125 PRINT TAB(40)B$
00135 NEXT A
00400 PRINT "IF YOU HAVE MORE VALUES FOR THIS PROGRAM"
00401 PRINT "TYPE 'YES'. IF YOU WISH TO GO TO THE MAIN LIST THEN"
00402 LINE INPUT "TYPE 'NO' ? ";M$
00403 IF M$="YES"THEN RETURN
00404 IF M$="NO"GOTO 505
00406 GOTO 400
00505 PRINT :INPUT "WHAT ITEM NUMBER FROM MAIN LIST ? ";Q
00507 IF Q>18GOTO 505
00508 IF Q<1GOTO 505
00510 ON Q GOTO 1000,1500,1900,2500,3000,3500,4000,4500,5000,1300
,1400
00515 ON Q-11GOTO 1450,1475,6000,1700,1800,1100,1200
00880 DATA "BLACK","BROWN","RED","ORANGE","YELLOW","GREEN"
00885 DATA "BLUE","VIOLET","GRAY","WHITE","777"
00900 DATA "1 PARALLEL RESISTORS","2 RESONANT FREQUENCY"
00905 DATA "3 SERIES CAPACITORS","4 REACTANCE","5 IMPEDANCE"
00910 DATA "6 ANTENNA LENGTH","7 OHMS LAW","8 POWER FORMULA"
00915 DATA "9 RESISTOR COLOR CODE","10 TRANSFORMER TURNS RATIO"
00920 DATA "11 DECIBELS","12 CONDUCTANCE","13 FREQUENCY TO TIME"
00925 DATA "14 TRANSMISSION LINE","15 ANT FIELD STRENGTH"
00930 DATA "16 PEAK RADAR PULSE POWER","17 PARALLEL INDUCTORS"
00935 DATA "18 PROGRAM CONVERSION TABLE","999"
01000 PRINT "THIS PROGRAM WILL CALCULATE PARALLEL RESISTORS"
01015 INPUT "HOW MANY RESISTORS ? ";R
01017 T=0:IF R>18GOTO 1015
01018 IF R<2GOTO 1015
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE < CONT'D >

```
01020 FOR X=1TO R:PRINT "RESISTOR VALUE ";X;" IN OHMS";
01022 INPUT " ? ";R1(X):IF X=R GOTO 1030
01025 NEXT X
01030 FOR X=1TO R:T=T+(1/R1(X)):IF X=R GOTO 1040
01032 NEXT X
01040 PRINT "TOTAL RESISTANCE IS ";1/T;" OHMS"
01050 GOSUB 400:PRINT :GOTO 1015
01100 PRINT "THIS PROGRAM WILL CALCULATE PARALLEL INDUCTORS"
01115 INPUT "HOW MANY INDUCTORS ? ";L
01117 T=0:IF L>18 GOTO 1115
01118 IF L<2GOTO 1115
01120 FOR X=1TO L:PRINT "INDUCTOR VALUE ";X;" IN HENRYS";
01122 INPUT " ? ";L1(X):IF X=L GOTO 1130
01125 NEXT X
01130 FOR X=1TO L:T=T+(1/L1(X)):IF X=L GOTO 1140
01132 NEXT X
01140 PRINT "TOTAL INDUCTANCE IS ";1/T;" HENRYS"
01150 GOSUB 400:PRINT :GOTO 1115
01200 PRINT "THIS PROGRAM WILL CONVERT ENTRIES (TO BE USED IN ";
01202 PRINT "MAIN PROGRAM) TO UNITS"
01204 PRINT "CONVERT ITEM"
01206 PRINT "(1)GIGA, (2)MEGA, (3)KILO, (4)MILLI, (5)MICRO, ";
01207 PRINT " (6)NANO, OR (7)PICO"
01208 INPUT "TO UNITS ? ";U
01210 IF U=1GOTO 1280
01212 IF U=2GOTO 1230
01214 IF U=3GOTO 1240
01216 IF U=4GOTO 1250
01218 IF U=5GOTO 1260
01220 IF U=6GOTO 1290
01222 IF U=7GOTO 1270
01230 INPUT "MEGA VALUE ? ";U
01234 PRINT "MEGA VALUE ";U;" IS ";U*1000000;" IN UNITS"
01236 GOSUB 400:PRINT :GOTO 1204
01240 INPUT "KILO VALUE ? ";U
01244 PRINT "KILO VALUE ";U;" IS ";U*1000;" IN UNITS"
01246 GOSUB 400:PRINT :GOTO 1204
01250 INPUT "MILLI VALUE ? ";U
01254 PRINT "MILLI VALUE ";U;" IS ";U*.001;" IN UNITS"
01256 GOSUB 400:PRINT :GOTO 1204
01260 INPUT "MICRO VALUE ? ";U
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE < CONT'D >

```
01264 PRINT "MICRO VALUE ";U;" IS ";U*1000001;" IN UNITS"
01266 GOSUB 400:PRINT :GOTO 1204
01270 INPUT "PICO VALUE ? ";U
01274 PRINT "PICO VALUE ";U;" IS ";U*10000000000001;" IN UNITS"
01276 GOSUB 400:PRINT :GOTO 1204
01280 INPUT "GIGA VALUE ? ";U
01284 PRINT "GIGA VALUE ";U;" IS ";U*1000000000;" IN UNITS"
01286 GOSUB 400:PRINT :GOTO 1204
01290 INPUT "NANO VALUE ? ";U
01294 PRINT "NANO VALUE ";U;" IS ";U*1000000001;" IN UNITS"
01296 GOSUB 400:PRINT :GOTO 1204
01300 PRINT "THIS PROGRAM WILL CALCULATE SECONDARY VOLTAGE"
01310 INPUT "PRIMARY VOLTAGE ? ";P
01315 INPUT "SECONDARY TURNS ? ";S
01320 INPUT "PRIMARY TURNS ? ";P1
01325 PRINT "THE SECONDARY VOLTAGE IS ";P*(S/P1)
01330 GOSUB 400:PRINT :GOTO 1310
01400 PRINT "THIS PROGRAM WILL CALCULATE DECIBEL RATIO OF TWO POW
ERS"
01415 INPUT "ENTER POWER ONE ? ";P1:INPUT "ENTER POWER TWO ? ";P2
01425 PRINT "THE DECIBEL VALUE IS ";10*(LOG(P1/P2)/LOG(10));" DB"
01430 GOSUB 400:PRINT :GOTO 1415
01455 PRINT "THIS PROGRAM WILL CALCULATE CONDUCTANCE IN MHOS"
01455 INPUT "WHAT IS THE RESISTANCE ? ";R
01456 IF R=0GOTO 1455
01460 PRINT "THE CONDUCTANCE IS ";1/R;" MHOS"
01465 GOSUB 400:PRINT :GOTO 1455
01475 PRINT "THIS PROGRAM WILL CALCULATE FREQUENCY TO TIME"
01476 INPUT "IS FREQUENCY IN HZ (1) OR KHZ (2) ? ";M
01477 IF M=1GOTO 1480
01478 IF M=2GOTO 1486
01479 GOTO 1476
01480 INPUT "FREQUENCY IN HZ ? ";F
01481 PRINT "THE FREQUENCY IS EQUAL TO ";1/F;" SECONDS":GOTO 1488
01486 INPUT "FREQUENCY IN KHZ ? ";F
01487 PRINT "THE FREQUENCY IS EQUAL TO ";1/(F/1000);" MICROSECOND
S"
01488 GOSUB 400:PRINT :GOTO 1476
01500 PRINT "THIS PROGRAM WILL CALCULATE RESONANT FREQUENCY"
01510 INPUT "INDUCTANCE IN HENRYS ? ";L
01515 INPUT "CAPACITANCE IN FARADS ? ";C
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE < CONT'D >

```
01520 P=2*(4*ATN(1)):P=P*(SQR(L*C)):P=1/P
01525 PRINT "THE RESONANT FREQUENCY IS ";P;" HZ"
01528 A=P/1000:B=A/1000:C=B/1000
01530 PRINT "IN KHZ";A;" IN MHZ";B;" IN GEGA HZ";C
01535 GOSUB 400:PRINT :GOTO 1510
01700 PRINT "THIS PROGRAM WILL CALCULATE FIELD STRENGTH"
01705 PRINT "AT MODERATE DISTANCES FROM THE TRANSMITTING ANTENNA"
01710 INPUT "WHAT IS THE EFFECTIVE ANTENNA HEIGHT IN METERS ? ";B
01715 INPUT "WHAT IS THE ANTENNA CURRENT IN AMPERES ? ";I
01720 INPUT "WHAT IS THE OPERATING WAVELENGTH IN METERS ? ";W
01725 INPUT "WHAT IS THE DISTANCE FROM ANTENNA IN METERS ? ";R
01730 E=(188*B*I)/(W*R)
01735 PRINT "THE FIELD STRENGTH IS ";E;" MILLIVOLTS PER METER"
01740 GOSUB 400:PRINT :GOTO 1710
01800 PRINT "THIS PROGRAM WILL CALCULATE THE PEAK"
01805 PRINT "POWER OF A RADAR PULSE"
01810 INPUT "WHAT IS THE PULSE WIDTH IN MICROSECONDS ? ";W
01815 INPUT "WHAT IS THE PULSE REPETITION RATE ? ";R
01820 INPUT "WHAT IS THE AVERAGE POWER IN WATTS ? ";P
01822 D1=1/R:W1=W/1000000
01825 D1=W1/D1
01830 P1=P/D1:REM PEAK POWER
01840 PRINT "THE PEAK POWER IS ";P1;" WATTS"
01845 PRINT "THE DUTY CYCLE IS ";D1
01850 GOSUB 400:PRINT :GOTO 1800
01900 PRINT "THIS PROGRAM WILL CALCULATE SERIES CAPACITORS"
01915 INPUT "HOW MANY CAPACITORS ? ";C
01917 T=0:IF C>18GOTO 1915
01918 IF C<2GOTO 1915
01920 FOR X=1TO C:PRINT "CAPACITOR VALUE ";X;" IN FARADS";
01922 INPUT " ? ";C1(X):IF X=C GOTO 1930
01925 NEXT X
01930 FOR X=1TO C:T=T+(1/C1(X)):IF X=C GOTO 1940
01932 NEXT X
01940 PRINT "TOTAL CAPACITANCE IS ";1/T;" FARADS"
01950 GOSUB 400:PRINT :GOTO 1915
02500 PRINT "THIS PROGRAM WILL CALCULATE INDUCTIVE AND CAPACITIVE
";
02505 PRINT "REACTANCE"
02510 INPUT "SOLVE FOR INDUCTIVE (1) OR CAPACITIVE (2) REACTANCE
? ";R
02515 IF R=2GOTO 2600
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE < CONT'D >

```
02520 PRINT "SOLVE FOR INDUCTIVE REACTANCE"
02525 INPUT "FREQUENCY IN HERTZ ? ";F
02530 INPUT "INDUCTANCE IN HENRYS ? ";L
02532 P=2*(4*ATN(1))
02535 PRINT "THE INDUCTIVE REACTANCE IS ";P*(F*L);" OHMS":GOTO 26
50
02600 PRINT "SOLVE FOR CAPACITIVE REACTANCE"
02610 INPUT "FREQUENCY IN HERTZ ? ";F
02615 INPUT "CAPACITANCE IN FARADS ? ";C
02625 PRINT "THE CAPACITIVE REACTANCE IS ";1/(3.1416*(F*C));" OHM
S"
02650 GOSUB 400:PRINT :GOTO 2510
03000 PRINT "THIS PROGRAM WILL CALCULATE SERIES OR PARALLEL IMPED
ANCE"
03010 INPUT "SERIES (1) OR PARALLEL (2) CIRCUIT ? ";Q
03020 IF Q=2GOTO 3307
03000 INPUT "RESISTANCE VALUE IN OHMS ? ";R
03065 INPUT "CAPACITANCE VALUE IN FARADS ? ";C
03070 INPUT "INDUCTANCE VALUE IN HENRYS ? ";L
03077 IF C=L THEN Z=R:GOTO 3200
03080 IF L>C GOTO 3100
03085 IF C>L GOTO 3150
03100 Z=L-C:Z=Z*Z:Z=Z+(R*R):Z=SQR(Z):GOTO 3200
03150 Z=C-L:Z=Z*Z:Z=Z+(R*R):Z=SQR(Z)
03200 PRINT "THE IMPEDANCE IS ";Z;" OHMS"
03215 GOTO 3340
03307 INPUT "RESISTANCE VALUE IN OHMS ? ";R
03308 INPUT "CAPACITANCE VALUE IN FARADS ? ";C
03309 INPUT "INDUCTANCE VALUE IN HENRYS ? ";L
03315 A=12/R:B=12/C:C=12/L
03320 IF B>C THEN X=B-C
03325 IF C>B THEN X=C-B
03327 T=A*A:X=X*X
03335 PRINT "THE IMPEDANCE IS ";12/(SQR(T+X));" OHMS"
03340 GOSUB 400:PRINT :GOTO 3010
03500 PRINT "THIS PROGRAM WILL CALCULATE ANTENNA LENGTH"
03505 INPUT "FREQUENCY IN MHZ ? ";F
03510 A=468/F:PRINT "THE HALF WAVE ANTENNA LENGTH IS ";A;" FEET";
03512 PRINT " OR ";:PRINT A*12;:PRINT " INCHES"
03514 PRINT "OPERATION WILL BE IN THE ";300/F;" METER BAND"
03600 GOSUB 400:PRINT :GOTO 3505
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE < CONT'D >

```
04000 PRINT "THIS PROGRAM WILL CALCULATE OHMS LAW"
04005 LINE INPUT "SOLVE FOR WHAT UNIT (E),(I), OR (R) ? ";U$
04006 IF U$="E"GOTO 4015
04008 IF U$="I"GOTO 4050
04010 IF U$="R"GOTO 4075
04012 GOTO 4005
04015 INPUT "CURRENT VALUE ? ";I
04006 INPUT "RESISTANCE VALUE ? ";R
04018 A=I*R:PRINT "THE VOLTAGE IS ";A:GOTO 4100
04000 INPUT "VOLTAGE VALUE ? ";E
04052 INPUT "RESISTANCE VALUE ? ";R
04055 A=E/R:PRINT "THE CURRENT IS ";A:GOTO 4100
04075 INPUT "VOLTAGE VALUE ? ";E
04076 INPUT "CURRENT VALUE ? ";I
04080 A=E/I:PRINT "THE RESISTANCE IS ";A
04100 GOSUB 400:PRINT :GOTO 4005
04500 PRINT "THIS PROGRAM WILL CALCULATE FOR POWER"
04505 INPUT "WHAT TWO UNITS (1)I&R, (2)E&R, (3)E&I ? ";U
04510 IF U=1GOTO 4550
04515 IF U=2GOTO 4575
04520 IF U=3GOTO 4600
04522 GOTO 4505
04550 INPUT "VALUE OF I,R ? ";A,B
04555 PRINT "POWER EQUALS ";B*(A*A);" WATTS":GOTO 4625
04575 INPUT "VALUE OF E,R ? ";A,B
04580 PRINT "POWER EQUALS ";B/(A*A);" WATTS":GOTO 4625
04600 INPUT "VALUE OF E,I ? ";A,B
04605 PRINT "POWER EQUALS ";A*B;" WATTS"
04625 GOSUB 400:PRINT :GOTO 4505
05000 PRINT "THIS PROGRAM WILL CALCULATE THE RESISTOR COLOR CODE"
05005 LINE INPUT "FIRST COLOR ? ";C1$
05008 RESTORE :A=0:FOR X=1TO 12
05016 READ Q$:IF Q#=C1$THEN A=A:GOTO 5070
05018 IF Q#="777"GOTO 5005
05020 A=A+10:NEXT X
05070 LINE INPUT "SECOND COLOR ? ";C2$
05072 RESTORE :B=0:FOR X=1TO 12
05078 READ R$:IF R#=C2$THEN B=B:GOTO 5125
05080 IF R#="777"GOTO 5070
05082 B=B+1:NEXT X
05125 LINE INPUT "THIRD COLOR ? ";C3$
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE < CONT'D >

```
05126 IF C3$="GOLD"THEN C=.1:GOTO 5200
05127 IF C3$="SILVER"THEN C=.01:GOTO 5200
05128 RESTORE :C=1:FOR X=1TO 12
05134 READ S$:IF S#=C3$THEN C=C:GOTO 5200
05136 IF S#="777"GOTO 5125
05142 C=C*10:NEXT X
05200 K=C*(A+B):PRINT "THE RESISTANCE IS ";K;" OHMS"
05205 S=K
05215 K=K/1000:PRINT "OR ";K;" KILO OHMS"
05225 K=K/1000:PRINT "OR ";K;" MEGA OHMS"
05235 K=K/1000:PRINT "OR ";K;" GIGA OHMS"
05300 PRINT "ENTRIES FOR NEXT QUESTION (NONE, SILVER, OR GOLD)"
05310 LINE INPUT "FOURTH COLOR ? ";C4$
05315 IF C4$="NONE"GOTO 5325
05318 IF C4$="SILVER"GOTO 5350
05320 IF C4$="GOLD"GOTO 5375
05325 M=.20*S:GOTO 5402
05350M=.10*S:GOTO 5402
05375 M=.05*S:GOTO 5402
05400 GOSUB 400:PRINT :GOTO 5005
05402 PRINT "THE RESISTOR TOLERANCE IS ";M
05404 PRINT "THE TOLERANCE RANGE IS ";S-M;" TO ";S+M;" OHMS":GOTO
5400
06000 PRINT "THIS PROGRAM WILL CALCULATE THE CHARACTERISTIC"
06005 PRINT "IMPEDANCE OF TRANSMISSION LINES"
06010 LINE INPUT "IS TRANSMISSION LINE TWO WIRE PARALLEL ? ";X$
06015 IF X$="YES"GOTO 6030
06020 LINE INPUT "IS TRANSMISSION LINE A COAXIAL LINE ? ";X$
06025 IF X$="YES"GOTO 6070
06028 PRINT :PRINT "TYPE 'YES' FOR ONE OF THE QUESTIONS":GOTO 601
0
06030 PRINT :PRINT "PARALLEL TRANSMISSION LINE"
06032 INPUT "WHAT IS THE SPACING BETWEEN CONDUCTORS IN INCHES ? "
;A
06034 INPUT "WHAT IS THE RADIUS OF ONE CONDUCTOR IN INCHES ? ";B
06036 Z=276*(LOG(A/B))
06038 PRINT :PRINT "THE IMPEDANCE IS ";Z;" OHMS"
06040 GOSUBB400:PRINT :GOTO 6010
06070 PRINT :PRINT "COAXIAL TRANSMISSION LINE"
06072 PRINT "WHAT IS THE INNER DIAMETER OF THE OUTER CONDUCTOR"
06074 INPUT "IN INCHES ? ";A
```



PROGRAM NAME: ELECTRONIC FORMULAS - CRABTREE < CONT'D >

```
06076 PRINT "WHAT IS THE OUTER DIAMETER OF THE INNER CONDUCTOR"
06078 INPUT "IN INCHES ? ";B
06080 Z=138*(LOG(A/B)):Z=Z*.675
06082 PRINT :PRINT "THE IMPEDANCE IS ";Z;" OHMS"
06084 GOSUB 400:PRINT :GOTO 6010
```

19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: METRIC/US CONVERSION - STILLMAN

```
00010 REM          METRIC <> US CONVERSION PROGRAM
00020 REM          R. STILLMAN
00030 PRINT :PRINT :GOSUB 860
00040 PRINT :PRINT :PRINT
00050 CLEAR :B=1:LINE INPUT " EXPRESSION : ";E$
00060 IF E$="" THEN PRINT :GOTO 50
00070 Y=1:FOR A=LEN(E$) TO 1 STEP -1:T=ASC(MID$(E$,A,1))-48
00080 IF T<10 AND T>=0 THEN X=X+(Y*T):Y=Y*10:NEXT A
00090 X$=CHR$(T+48):IF X$="" GOTO 50
00100 IF X$="U" THEN GOSUB 1300:PRINT :GOTO 50
00110 IF X$="O" GOTO 470
00120 IF X$="+" THEN B=1:S=X:X=0:Y=1:GOTO 330
00130 IF X$="-" THEN B=2:S=X:X=0:Y=1:GOTO 330
00140 IF X$="*" THEN B=3:S=X:X=0:Y=1:GOTO 330
00150 IF X$="/" THEN B=4:S=X:X=0:Y=1:GOTO 330
00160 IF X$="." THEN X=X/Y:Y=1:NEXT A
00170 IF X$="M" AND M=8 THEN M=1:U$="MM":F=1:GOTO 300
00180 IF X$="C" THEN M=2:U$="CM":F=1:GOTO 300
00190 IF X$="T" THEN M=3:U$="MTR":F=1:GOTO 300
00200 IF X$="M" AND M=3 THEN F=1:GOTO 300
00210 IF X$="K" THEN M=4:U$="KM":F=1:GOTO 300
00220 IF X$="I" THEN M=5:U$="IN":F=1:GOTO 300
00230 IF X$="F" THEN M=6:U$="FT":F=1:GOTO 300
00240 IF X$="Y" THEN M=7:U$="YD":F=1:GOTO 300
00250 IF X$="M" THEN M=8:U$="MI":F=1:GOTO 300
00260 IF X$="=" THEN W=M:W$=U$:F=0:M=0:IF X>0 GOTO 440
00270 IF X$="L" GOTO 450
00280 IF A=0 GOTO 360
00290 IF X>0 AND F=1 GOTO 440
00300 IF X>0 AND F=1 GOTO 440
00310 IF A>0 THEN NEXT A
00320 GOTO 360
00330 C=M:F=0:IF M>4 AND B>2 THEN M=3
00340 IF M=0 AND W>3 THEN M=3
00350 ON M GOSUB 620,630,640,650,660,670,680,690:M=0:NEXT A
00360 IF M=0 GOTO 440
00370 ON M GOSUB 700,710,720,730,740,750,760,770
00380 IF B>2 AND C>0 GOTO 440
00390 ON B GOSUB 480,490,500,560
00400 IF B>2 AND C>0 GOTO 440
00410 IF W=0 GOTO 440
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: METRIC/US CONVERSION - STILLMAN < CONT'D >

```
00420 ON W GOSUB 780,790,800,810,820,830,840,850
00430 PRINT TAB(355;E;TAB(45);W#;PRINT :PRINT :GOTO 50
00440 PRINT " SORRY, EXPRESSION : ";E#;" IS NOT LEGAL":PRINT :PR
INT :GOTO 50
00450 PRINT " SORRY, PLEASE USE 'MI' FOR MILE AND 'MM' FOR
MILLIMETER"
00460 PRINT :PRINT :GOTO 50
00470 PRINT " SORRY, LETTER 'O' CANNOT BE USED":PRINT :PRINT :GOT
O 50
00480 E=S+X:RETURN
00490 E=X-S:RETURN
00500 E=S*X:RETURN
00510 IF W=2 THEN E=(S*X)*100
00520 IF W=3 THEN E=S/X
00530 IF W=4 THEN E=(S*X)/1000
00540 IF W>4 THEN E=X*S
00550 RETURN
00560 E=X/S:RETURN
00570 IF W=2 THEN E=(X/S)/100
00580 IF W=3 THEN E=X/S
00590 IF W=4 THEN E=(X/S)*1000
00600 IF W>4 THEN E=X/S
00610 RETURN
00620 S=S/1000:RETURN
00630 S=S/100:RETURN
00640 S=S:RETURN
00650 S=S*1000:RETURN
00660 S=S*.0254:RETURN
00670 S=S*.3048:RETURN
00680 S=S*.914401:RETURN
00690 S=S*1609.34:RETURN
00700 X=X/1000:RETURN
00710 X=X/100:RETURN
00720 X=X:RETURN
00730 X=X*1000:RETURN
00740 X=X*.0254:RETURN
00750 X=X*.3048:RETURN
00760 X=X*.914401:RETURN
00770 X=X*1609.34:RETURN
00780 E=E*1000:RETURN
00790 E=E*100:RETURN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: METRIC/US CONVERSION - STILLMAN < CONT'D >

```
00800 E=E:RETURN
00810 E=E/1000:RETURN
00820 E=E/.0254:RETURN
00830 E=E/.3048:RETURN
00840 E=E/.914401:RETURN
00850 E=E/1609.347:RETURN
00860 PRINT TAB(20)* CONVERSION PROGRAM *:PRINT
00870 PRINT TAB(20)"METRIC <> U.S. - LENGTHS":PRINT :PRINT
00880 FOR A=1 TO 600:NEXT A
00890 PRINT " PROGRAM WILL CONVERT METRIC TO METRIC UNIT LENGTHS
, METRIC TO "
00900 PRINT "U.S. EQUIVELENT, U.S. LENGTH TO U.S. LENGTHS, U.S. L
ENGTHS TO "
00910 PRINT "METRIC EQUIVELENTS."
00920 PRINT " LENGTHS IN ONE UNIT, EITHER U.S. OR METRIC MAY BE
ADDED OR "
00930 PRINT "SUBTRACTED WITH THE RESULT BEING EXPRESSED IN EITHER
U.S. OR "
00940 PRINT "METRIC EQUIVELENTS.":PRINT
00950 PRINT TAB(10)"EXAMPLE :";TAB(25)"2FT+15CM=MTR"
009600PRINT TAB(25)"12.32IN-1.25CM=MM":PRINT :
00970 PRINT "TO CONTINUE TYPE 'CR'":PAUSE
00980 PRINT :PRINT :PRINT :PRINT
009900PRINT " LENGTHS IN ANY UNIT EITHER U.S. OR METRIC MAY ALSO
BE DIVIDED"
01000 PRINT "OR MULTIPLIED WITH THE RESULTS BEING EXPRESSED IN AN
Y U.S. OR "
01010 PRINT "METRIC UNIT.":PRINT
01020 PRINT TAB(10)"EXAMPLE :";TAB(25)"12.91MTR/3=CM"
01030 PRINT TAB(25)"14FT*7=MTR":PRINT :PRINT
01040 PRINT "TO CONTINUE TYPE 'CR'":PAUSE
01050 PRINT :PRINT :PRINT :PRINT
01060 PRINT " DIRECT UNIT CONVERSION WITHIN METRIC,U.S., OR BET
WEEN SYSTEMS"
010700PRINT "MAY ALSO BE ACCOMPLISHED.":PRINT
01080 PRINT TAB(10)"EXAMPLE : ";TAB(25)"19.3MM=IN"
01090 PRINT TAB(25)"100KM=MI"
01100 PRINT :PRINT :PRINT
01110 PRINT "TO CONTINUE TYPE 'CR'":PAUSE
01120 PRINT :PRINT :PRINT
01130 PRINT " THERE ARE EIGHT LEGAL TERMS USED TO DESCRIBE UNITS
AND FOUR"
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: METRIC/US CONVERSION - STILLMAN

```
01140 PRINT "MATHEMATICAL FUNCTIONS. ON REQUEST FOR AN EXPRESSION
A NUMBER IS"
01150 PRINT "ENTERED, THEN ITS UNITS, A MATH FUNCTION IF REQUIRED
, AN EQUAL"
01160 PRINT "SIGN, AND THE UNIT ANSWER DESIRED. U.S. UNITS LESS
THAN WHOLE"
01170 PRINT "NUMBERS ARE ENTERED IN TENTHS. (1/4 WOULD BE ENTERED
0.25IN)"
01180 PRINT
01190 PRINT " FOLLOWING ARE A LIST OF LEGAL UNITS USED IN THIS P
ROGRAM. TO"
01200 PRINT "REVIEW THEM AT ANY TIME ENTER 'UNITS' ON PROMPT 'EXP
RESSION'"
01210 PRINT :PRINT "TO CONTINUE TYPE 'CR'";:PAUSE
01220 PRINT :PRINT :PRINT
01230 PRINT " U.S. FRACTIONS MAY BE CONVERTED TO THEIR DECIMAL E
QUIVALENTS BY"
01240 PRINT "ENTERING ONE (1) AND THE UNIT 'IN' DIVIDED BY THE FR
ACTIONAL "
01250 PRINT "DENOMINATOR FOLLOWED BY EQUALS (=) INCH. THE RESULT
S MAY ALSO "
01260 PRINT "BE CONVERTED DIRECTLY TO ANY OF THE OTHER LEGAL UNIT
S ":PRINT
01270 PRINT TAB(10)"EXAMPLES :";TAB(25)"1IN/8=IN - DECIMAL 1/8
"
01280 PRINT TAB(25)"1IN/4=MM - MM EQUIVALENT 1/4 IN"
01290 PRINT :PRINT :PRINT "TO CONTINUE TYPE 'CR'";:PAUSE
01300 PRINT :PRINT :PRINT
01310 PRINT TAB(5)"LEGAL UNITS :":PRINT
01320 PRINT TAB(10)"IN = INCH";TAB(35)"MM = MILLIMETER"
01330 PRINT TAB(10)"FT = FOOT";TAB(35)"CM = CENTIMETER"
01340 PRINT TAB(10)"YD = YARD";TAB(35)"MTR = METER"
01350 PRINT TAB(10)"MI = MILE";TAB(35)"KM = KILOMETER"
01360 PRINT :PRINT TAB(10)"+" = PLUS";TAB(35)"*" = MULTIPLY"
01370 PRINT TAB(10)"-" = MINUS";TAB(35)"/" = DIVIDE"
01380 PRINT :PRINT "TO START TYPE 'CR'";:PAUSE :PRINT
01390 GOSUB 1410:PRINT :PRINT " EXAMPLE - EXPRESSION : 12.3MM+5
7MM=CM"
01400 GOSUB 1410:PRINT :PRINT :RETURN
01410 PRINT :FOR A=1TO 75:PRINT "-";:NEXT A:PRINT :RETURN
01420 REM
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: METRIC/US CONVERSION - STILLMAN < CONT'D >

```
01430 REM " ROBERT F STILLMAN"
01440 REM " 9709 HILLSHIRE DR"
01450 REM " RICHMOND ILL. 60071"
01460 REM " (815) 678-4374"
```

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	00
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



# I N D E X

## Volume II

TED8 Source Code 885-1014

<u>Program Name</u>	<u>Page Number</u>
Renumber-Ignatius	1
Renumber/Merge-Moss	22
Utility Routines-Price	43
TAPE Management-Turner	70
8080 Disassembler-Shiffrin	108
CCS'SUPPRESS-Morgan	121
Mail Label-Farmer	130





<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS

```
TITLE      'EXTENDED B.H. BASIC RENUMBER OPTION  VER. 1.0'
STL        'INSTRUCTIONS & DESCRIPTION'
**** EXTENDED B.H. BASIC RENUMBER OPTION
** WRITTEN BY  MARK J. IGNATIUS
*              LAKEWOOD, OHIO
*              AUGUST 1978
*
* -INSTRUCTIONS AND DESCRIPTION-
* -MAKE COPIES FOR READY REFERENCE-
* -SEE PAGE 3 FOR ASSEMBLY INSTRUCTIONS-
*
* VER. 1.0
*
** THIS PROGRAM, WHEN RUN ALONG WITH EXTENDED
* B.H. BASIC, WILL ALLOW THE USER TO
* RENUMBER HIS PROGRAM STARTING WITH ANY LINE
* NUMBER AND IN ANY INCREMENT. THIS INCLUDES
* LINE NUMBER REFERENCES WITHIN THE TEXT.
*
** THIS PROGRAM RESIDES IN HIGH MEMORY AND IS
* SELF-CONFIGURING (IT WILL CONFIGURE HIGH
* MEMORY AUTOMATICLY).
*
* THIS PROGRAM 'MUST' RESIDE IN HIGH MEMORY.
* DO NOT RUN A VERSION OF THIS PROGRAM CONFIGURED
* FOR LESS MEMORY THAN IN YOUR SYSTEM. IMPROPER
* OPERATION WILL RESULT.
*
* THIS PROGRAM CAN BE LOADED AFTER BASIC HAS
* BEEN UP & RUNNING. THERE IS NO NEED TO
* RELOAD BASIC.
*   THIS PROGRAM MUST BE STARTED BY PUSHING GO
* AFTER LOADING.
*
* *** CAUTION ***
* LOADING THIS PROGRAM SCRATCHES USER PROGRAM.
* SAVE BASIC PROGRAM BEFORE LOADING.
*
*** SYNTAX:
*
* RENUMBER      START AT LINE 10, INCREMENT OF 10
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

```
*      RENUMBER X      START AT LINE X, INCREMENT OF 10
*      RENUMBER X,Y    START AT LINE X, INCREMENT OF Y
*
**    THIS COMMAND IS ENTERED IN THE COMMAND MODE
*    AS ANY OTHER COMMAND (EG. BUILD, LIST, RUN,
*    ETC.), WITH COMMAND COMPLETION.
*
**    ***** SPECIAL NOTE *****
*    THIS PROGRAM REMOVES THE 'CONTINUE' COMMAND.
*    IT WILL NO LONGER BE VALID. TO SIMULATE
*    CONTINUE, USE THE 'STEP' COMMAND.
*
      EJECT
***  -ERROR REPORTING-
*
*    WHEN THE COMMAND IS ENTERED AND THE CARRIAGE
*    RETURN HIT, THE COMMAND IS CHECKED FOR
*    VALID EXPRESSIONS. IF EITHER EXPRESSION
*    EVALUATES TO 0 OR 65535, OR THE RENUMBERING
*    CAUSES AN OVERFLOW OVER 65535, AN ERROR
*    MESSAGE IS GENERATED. THIS IS DONE PRIOR
*    TO THE ACTUAL RENUMBERING PROCESS SO NEW
*    VALUES MAY BE ENTERED.
*
*    ONCE THE RENUMBERING PROCESS BEGINS
*    (THE MESSAGE 'STANDBY' IS OUTPUT TO THE
*    CONSOLE), INTERRUPTS FROM THE CONSOLE ARE
*    DISABLED.
*
**    ***** SPECIAL NOTE *****
*    IF THE PROGRAM IS INTERRUPTED (STOPPED) DURING
*    THE RENUMBERING PROCESS, THE USER PROGRAM WILL
*    BE DESTROYED AND WILL NOT FUNCTION CORRECTLY.
*    IF THE PROGRAM MUST BE STOPPED, RETURN TO THE
*    MONITOR AND DO A 'COLD' START AT 040.100.
*    THIS WILL OF COURSE SCRATCH THE USER PROGRAM,
*    BUT WILL NOT AFFECT BASIC OR THE RENUMBER
*    OPTION.
*
**    DURING THE RENUMBERING PROCESS, IF AN ERROR
*    IS FOUND, SUCH AS A MISSING LINE NUMBER
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

```
*      (EG. GOTO 1000, WHERE LINE 1000 DOES NOT
*      EXIST), AN ERROR MESSAGE IS DISPLAYED
*      ON THE CONSOLE IN THE FORMAT:
*      ERR - LN# 500
*      THIS MEANS THAT LINE 500 WAS FOUND TO
*      HAVE AN ERROR IN IT. THE LINE NUMBER 500
*      IS THE LINE NUMBER OF THE PROGRAM 'BEFORE'
*      IT WAS RENUMBERED. ONLY THAT PART OF THE
*      LINE FOUND TO BE IN ERROR IS IGNORED,
*      THE REST OF THE LINE WILL BE FIXED.
*      THE SAME LINE NUMBER MAY BE LISTED MORE
*      THAN ONCE, AND ALL LINES WITH ERRORS
*      WILL BE LISTED.
*
**     IF, WHEN WORKING WITH A LARGE PROGRAM,
*     A LINE IS CHANGED TO CONTAIN MORE
*     CHARACTERS SO THAT A MEMORY OVERFLOW
*     WOULD RESULT, THE WHOLE LINE WILL BE IGNORED
*     AND AN ERROR MESSAGE WILL BE GENERATED
*     IN THE FORMAT EXPLAINED ABOVE.
*     ALSO, ANY CHANGE THAT WOULD CAUSE THE LINE
*     TO EXCEED 99 CHAR'S IS IGNORED AND AN ERROR
*     IS FLAGGED (AS ABOVE). SOME LINES MAY EXCEED
*     80 CHAR'S (81-99) AND NOT BE FLAGGED AS
*     ERRORS. THESE LINES ARE OK AND WILL FUNCTION
*     CORRECTLY.
*
      STL      'ASSEMBLY INSTRUCTIONS'
      EJECT
**** *****
*
**** ASSEMBLY INSTRUCTIONS
*
**     ALL THAT IS NECESSARY TO ASSEMBLE THIS
*     PROGRAM IS TO EQUATE TWO LABELS.
*     +++ NO OTHER UNDERSTANDING OF THIS
*     +++ PROGRAM IS NECESSARY.
*
**     -FIRST LABEL-
*     THE FOLLOWING LABEL ($MLMIT) IS TO BE SET TO
*     THE HIGHEST MEMORY ADDRESS IN YOUR SYSTEM.
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

\* EXAMPLES:

\* IF YOU HAVE 24K, SET EQU TO 177377A

\* IF YOU HAVE 16K, SET EQU TO 137377A

\*

\* -HERE'S THE LABEL-

\$MLMIT EQU 137377A FOR 16K OF MEMORY

\* THE PROGRAM WILL LOCATE ITSELF AND  
\* CONFIGURE HIGH MEMORY BASED ON THE  
\* ABOVE LABEL. THIS PROGRAM WILL START  
\* APPROXIMATELY 1K BELOW THE ABOVE LABEL.  
\* (ACTUAL PROGRAM IS 792 BYTES IN LENGTH)

\*

\*\* -SECOND LABEL-

\* THE FOLLOWING LABEL (\$VERS) WILL DIRECT  
\* THE ASSEMBLER TO ASSEMBLE FOR THE VERSION  
\* OF EXT.D. B.H. BASIC DESIRED.

\*

\* FOR VERSION 10.01.01 SET EQU TO 0

\* FOR VERSION 10.02.01 SET EQU TO 1

\*

\* -HERE'S THE LABEL-

\$VERS EQU 0 FOR 10.01.01

\*\*\* AFTER THE ABOVE TWO LABELS HAVE BEEN  
\* CHANGED, THE PROGRAM SHOULD BE SAVED  
\* THEN ASSEMBLED.

\*

\*\*\* \*\* NOTE \*\*

\* THIS PROGRAM IS SET UP SO THAT ALL DATA IS  
\* LISTED ON THE LIST DEVICE WHETHER ASSEMBLED  
\* OR NOT. DATA NOT ASSEMBLED WILL NOT HAVE DATA  
\* ON THE LEFT SIDE OF THE LISTING.

\*

\*

\*

\* STL 'EQUIVALENCES AND SETUP'

\* EJECT

\*\*\*\*\* START OF PROGRAM



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

\*

ORG \$MLMIT-1024

\*\*\* THE FOLLOWING 2 INSTRUCTIONS ARE FOR  
\* LISTING AND ASSEMBLY CONTROL

LON I LIST ALL DATA  
IF \$VERS TURN OFF IF \$VERS=1

\*\*\* EQUIVALENCES FOR 10.01.01  
\* ARE ASSEMBLED IF \$VERS = 0  
\* OTHERWISE THEY ARE SKIPPED  
\*

\$BRADR	EQU	042121A	BRANCH ADDRESS
\$CMWRD	EQU	063162A	COMMAND WORD ADDRESS
START	EQU	040100A	START OF BASIC
GETVALU	EQU	053315A	GET VALUE FROM COMMAND
ERR.CC	EQU	064236A	ERROR - CONTROL C
ERR.IN	EQU	064274A	ILLEGAL NUMBER
ERR.IU	EQU	064302A	ILLEGAL USE
ERR.OV	EQU	064317A	OVERFLOW
REPLACE	EQU	065153A	REPLACE LINE ROUTINE
IFLT	EQU	067264A	FLOAT NUMBER
TESTCHR	EQU	070121A	TEST FOR VALUE INPUT
TDI	EQU	071307A	TYPE DECIMAL INTEGER
FTA	EQU	076336A	FLOATING POINT TO ASCII
CHARCNT	EQU	077372A	COUNT CHARACTERS
CRLF	EQU	100011A	CARRIAGE RET - LINE FEED
GDIGIT	EQU	100027A	GET A DIGIT
S\$CSIC	EQU	101122A	SET \$CSIC INTERRUPT
OLDEND	EQU	102331A	OLD PROGRAM END
HIGHMEM	EQU	102362A	CONF'D HIGH MEMORY
COMMSTR	EQU	102372A	COMMAND STRING
BUFFER	EQU	103136A	TEMPORARY STORAGE
PROGRAM	EQU	103303A	START OF BASIC PROGRAM

EJECT

\*\*\*\*\* THE FOLLOWING DIRECTIVE CHANGES THE  
\* ASSEMBLY FLAG  
\* IF VERSION 10.01.01 WAS BEING ASSEMBLED.

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

\* THE FOLLOWING WILL NOT BE. IF 10.01.01  
\* WAS NOT BEING ASSEMBLED, THE FOLLOWING  
\* WILL BE.

ELSE

EJECT

\*\*\*\*\*  
\* EQUIVALENCES FOR 10.02.01  
\* ARE ASSEMBLED IF \$VERS = 1  
\* OTHERWISE THEY ARE SKIPPED  
\*

\$BRADR	EQU	042123A	BRANCH ADDRESS
\$CMWRD	EQU	063360A	COMMAND WORD ADDRESS
START	EQU	040100A	START OF BASIC
GETVALU	EQU	054032A	GET VALUE FROM COMMAND
ERR.CC	EQU	065105A	ERROR - CONTROL C
ERR.IN	EQU	065143A	ILLEGAL NUMBER
ERR.IU	EQU	065151A	ILLEGAL USE
ERR.OV	EQU	065175A	OVERFLOW
REPLACE	EQU	066054A	REPLACE LINE ROUTINE
IFLT	EQU	070376A	FLOAT NUMBER
TESTCHR	EQU	071307A	TEST FOR VALUE INPUT
TDI	EQU	074046A	TYPE DECIMAL INTEGER
FTA	EQU	101355A	FLOATING POINT TO ASCII
CHARCNT	EQU	103011A	COUNT CHARACTERS
CRLF	EQU	103030A	CARRIAGE RET - LINE FEED
GDIGIT	EQU	103046A	GET A DIGIT
\$CSIC	EQU	104040A	SET \$CSIC INTERRUPT
OLDEND	EQU	105253A	OLD PROGRAM END
HIGHMEM	EQU	105304A	CONF'D HIGH MEMORY
COMMSTR	EQU	105314A	COMMAND STRING
BUFFER	EQU	106060A	TEMPORARY STORAGE
PROGRAM	EQU	106226A	START OF BASIC PROGRAM

\*\*\*\*\*  
\* THE FOLLOWING DIRECTIVE CAUSES THE REST  
\* OF THE PROGRAM TO BE ASSEMBLED REGARDLESS  
\* OF ANY ASSEMBLY FLAGS.

ENDIF  
EJECT

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

\*\*\*\*\* INITIALIZE APPROPRIATE BASIC VERSION

\*

\* TRANSFER NEW COMMAND WORD TO BASIC

```
ENTRY  LXI    H,$CMWRD  COMMAND WORD LOCATION
        LXI    D,RNMSG  NEW COMMAND WORD
        MVI    B,8      8 CHARACTERS
ENTY    LDAX   D         GET A CHAR
        MOV    M,A      MOVE IT
        INX    H        BUMP POINTERS
        INX    D
        DCR    B        DONE?
        JNZ    ENTY     NOT YET!
```

\* SET BRANCH ADDRESS TO THIS PROGRAM

```
LXI     H,RENUMB  ADDR OF THIS PROGRAM
SHLD    $BRADR   MOVE IT TO BASIC
```

\*\* CONFIGURE HIGH MEMORY

\$MSET EQU \*\*16 NEW MEMORY LIMIT

```
LXI     H,$MSET  GET LIMIT
SHLD    HIGHMEM  SET IT!
```

\*\*\*\*\* GO TO BASIC \*\*\*\*\*

```
JMP     START    COLD START
```

\* RENUMBER COMMAND WORD

RNMSG DB 'RENUMBER'

```
STL     'RENUMBER OPTION STORAGE AND DATA'
EJECT
```

\* RENUMBER OPTION STORAGE & DATA

```
TEN     EQU    10      DEFAULT VALUE
$RET    EQU    040110A  ADDR OF RETURN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

RSTRT	EQU	040103A	RETURN TO BASIC
INCREM	DW	TEN	INCREMENT
LINNO	DW	TEN	STARTING LINE NO.
CURLIN	DW	TEN	CURRENT LINE NO.
PNTRA	DS	2	CURRENT LINE POINTER
PNTRB	DS	2	INTERNAL LINE POINTER
CFLAG	DB	','	COMMA FLAG
CHRCNT	DB	0	REPLACEMENT STRING COUNT
OLDCNT	DB	0	OLD LINE CHAR. COUNT
ERRCNT	DB	0	ERROR COUNT

```

      STL      'INITIALIZATION & OPENING TEST'
      EJECT
****      START OF RENUMBER OPTION      ****
*          COLLECT INPUT EXPRESSIONS (IF ANY)

```

```

RENUMB  CALL    S$CSIC      SET CTL-C INTERRUPT
        DW      ERR.CC
        LXI     D,TEN      X=10 DEFAULT
        PUSH    D          SAVE Y (=10)
        CALL    TESTCHR    TEST FOR X ENTRY
        ANA     A          AN ENTRY?
        JZ      NUMTST     NO!
        CALL    GETVALU    YES, GET IT
        POP     H          GET Y
        PUSH    D          SAVE X
        CALL    TESTCHR    TEST FOR Y ENTRY
        ANA     A          ENTRY?
        JZ      NUMTT     NO Y ENTRY
        RST     6          SET FOR NEXT GET
        DB      2720
        CALL    GETVALU    GET Y VALUE

```

```

**      TEST INPUT PARAMETERS
*      (DE)=Y INCREMENT
*      (STACK)= STARTING LINE NO.

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

NUMTST	XCHG		<HL>=INCREMENT
	CALL	ILNTST	IS INCREMENT OK?
	JZ	ERR.IN	NO
NUMTT	SHLD	INCREM	SAVE IT
	MOV	B,H	MOVE IT TO <BC>
	MOV	C,L	
	POP	H	GET STARTING LINE NO.
	CALL	ILNTST	IS LINE NO. OK?
	JZ	ERR.IN	NO
	SHLD	LINNO	SAVE IT
	LXI	D,PROGRAM	<DE>=TEXT POINTER
	CALL	ENDTST	TEST OF EXISTING PROGRAM
	JZ	ERR.IU	ILLEGAL USE, NO PROGRAM
NUTST1	CALL	FNDZRO	MOVE TO NEXT LINE
	DAD	B	NEXT NEW LN #
	JC	ERR.OV	LN # OVERFLO
	CALL	ILNTST	TEST NEW LN #
	JZ	ERR.IN	ILLEGAL LN #
	CALL	ENDTST	TEST FOR FINISH
	JNZ	NUTST1	NOT YET

\* TEST IS OK, SEND MESSAGE AND GO

```

RST      7
DB       7,'STANDBY'
CALL     CRLF
CALL     S$CSIC      DISABLE CONSOLE INT
DW       $RET
JMP      RENUGO      GO TO IT!!!

```

STL 'INITIAL SUBROUTINES'

EJECT

\* TEST FOR LN # = TO 377.377

ENDTST	LDAX	D
	INX	D
	CPI	377Q
	JNZ	ENDT1
	LDAX	D
	CPI	377Q
ENDT1	INX	D

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

RET

\* TEST FOR (HL)= 000.000 OR 377.377

```
ILNTST  MOV    A,H
        ANA    A          (H)=0?
        JNZ    ILNT      NO
        MOV    A,L
        ANA    A          (L)=0?
        RZ      (HL)=0
ILNT     MOV    A,H
        INR    A          (H)=377?
        RNZ    NO
        MOV    A,L
        INR    A          (L)=377?
        RET
```

\* MOVE TO END OF LINE

```
FNDZRO  LDAX   D          GET CHAR
        INX    D
        CPI    0          EOL YET?
        JNZ    FNDZRO     NO
        RET          AT END!
```

STL 'RENUMBER CONTROL'  
EJECT

\* RENUMBER CONTROL ROUTINE

\* ALL DATA SAVED

\* REGISTERS FREE TO USE

\*

```
RENUGO  XRA    A
        STA    ERRCNT     CLEAR ERROR COUNT
        LXI    H,PROGRAM  POINT TO FIRST LN #
RENCON  MOV    E,M        (DE)=LN #
        INX    H
        MOV    D,M
        INX    H
        XCHG
        CALL   ILNT       TEST FOR END
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

	XCHG		
	JZ	RENULN	GO RENUMBER THE LINES
	CALL	CODETST	TEST FOR CODE IN LINE
	JZ	RENCON	NO CODE, DO MORE
	SHLD	PNTRA	SAVE FOR LATER
RENC	LXI	B, COMMSTR	COMMAND STRING POINTER
	MOV	A, M	MOVE TEXT LINE TO
	STAX	B	COMMAND BUFFER
	INX	H	
	INX	B	
	ANA	A	DONE?
	JNZ	RENC	NOT YET
	XCHG		
	SHLD	CURLIN	SAVE THE LN #
	CALL	MODIFY	FIX THE LINE IN COMMSTR
	JNZ	SKIP	LINE WON'T FIT IN MEMORY
	LHLD	CURLIN	RESTORE LN #
	XCHG		
	LXI	H, COMMSTR	POINT TO COMMAND BUFFER
SKIP	CALL	REPLACE	REPLACE THE LINE IN TEXT
	LHLD	PNTRA	LN # POINTER
	XCHG		
	CALL	FNDZRO	MOVE TO END OF CHANGED LN
	XCHG		
	JMP	RENCON	DO ANOTHER LINE
	STL	'CODE TEST'	
	EJECT		
*			TEST FOR CODE IN CURRENT LINE
CODETST	PUSH	D	SAVE LN #
	PUSH	H	SAVE POINTER
CODET	MOV	A, M	GET CHAR
	ANA	A	TEST IT
	JZ	NOCOD	END OF LINE
	JP	CODM	NOT A CODE
	CPI	2170	
	JZ	YESCOD	ITS A GOSUB
	CPI	2200	
	JZ	YESCOD	ITS A GOTO
	CPI	2120	

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

```

        JZ      CNTRT      ITS A CNTRL
        CPI     2610
        JZ      THLS      ITS A THEN
        CPI     2230
        JZ      THLS      ITS A LIST
CODM    INX      H         NO CODE YET
        JMP     CODT      TRY ANOTHER ONE

*      FOUND CNTRL, TEST FOR 0
CNTRT   CALL    WEEDSP     WEED OUT SPACES
        CPI     0600      IS IT 0?
        JZ      YESCOD     YES
        JMP     CODM      NO

*      FOUND THEN OR LIST, LOOK FOR # TO FOLLOW
THLS    CALL    WEEDSP     WEED OUT SPACES
        CPI     0600
        JC      CODM      < '0'
        CPI     0720
        JC      YESCOD     YES A #
        JMP     CODT

        EJECT
*      WEED OUT SPACES
WEEDSP  INX      H
WEDSP   MOV     A,M
        CPI     ' '      IS IT SPACE?
        JZ      WEEDSP     YES, KEEP MOVING
        RET             NO, DONE

*      NO CODE FOUND IN THIS LINE
NOCOD   POP      D         DISCARD OLD POINTER
        POP      D         DISCARD LN #
        INX      H         BUMP POINTER
        XRA      A         CLEAR FLAG
        RET

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

\* CODE FOUND

```
YESCOD POP      H      RESTORE OLD POINTER
        POP      D      RESTORE LN #
        MVI      A,1    SET FLAG
        ANA      A
        RET
```

```
STL     'LINE RENUMBER AND EXIT'
EJECT
```

\* TEXT MODIFICATION DONE  
\* RENUMBER THE LINES

```
RENULN  LHL D      INCREM
        MOV      B,H      INCREMENT TO (BC)
        MOV      C,L
        LHL D      LINNO   LINE NO. TO (HL)
        LXI      D,PROGRAM (DE)= PROGRAM POINTER
RENU     CALL     ENDTST   TEST FOR END
        JZ       RENDON   YES, EXIT
        DCX      D        NO, BACK UP
        DCX      D
        XCHG      (DE)=LN #, (HL)=PNTR
        MOV      M,E      MOVE NEW LN # TO MEMORY
        INX      H
        MOV      M,D
        INX      H
        XCHG      (DE)=PNTR, (HL)=LN #
        DAD      B        ADD INCREMENT
        CALL     FNDZRO   MOVE TO END OF LINE
        JMP      RENU     DO MORE
```

\*\*\*\* RENUMBER OPTION COMPLETE  
\*\*\*\* RETURN TO BASIC

```
RENDON  CALL     CRLF
        RST      7        SEND MESSAGE
        DB       16,'RENUMBER COMP. '

        LDA      ERRCNT   GET ERROR COUNT
        ANA      A        ANY ERRORS?
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

	JZ	EXIT	NO
	PUSH	PSW	YES, SAVE THEM
	RST	7	SEND MESSAGE
	DB	9, 'ERRORS = '	
	POP	PSW	GET ERRORS
	MOV	E, A	NO. OF ERRORS TO <DE>
	MVI	D, 0	
	CALL	TDI	PRINT NO. OF ERRORS
EXIT	CALL	CRLF	
	CALL	CRLF	
**	RETURN TO BASIC		
	JMP	RSTRT	RETURN
	STL	'LINE MODIFICATION'	
	EJECT		
**	MODIFY THE LINE		
MODIFY	LXI	H, COMMSTR	POINT TO COMMAND STRING
	CALL	CHARCNT	COUNT CHAR'S
	STA	OLDCNT	SAVE COUNT
	DCX	H	
MODFCO	INX	H	BUMP POINTER
MODFC	MOV	A, M	GET A CHAR
	ANA	A	0?
	JZ	MODFDN	YES, LINE FINISHED
	JP	MODFCO	NOT A CODE, CONTINUE
	CPI	2120	
	JZ	CNTRL0	HAVE CNTRL CODE
	CPI	2170	
	JZ	MOTONO	HAVE GOSUB
	CPI	2200	
	JZ	MOTONO	HAVE GOTO
	CPI	2610	
	JZ	MOTONO	HAVE THEN
	CPI	2230	
	JZ	MOTONO	HAVE LIST
	JMP	MODFCO	

<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

\* HAVE A CONTROL CODE

```
CNTRL0  CALL    WEEDSP    WEED OUT SPACES
        CPI     '0'      IS IT 0?
        JNZ     MODFC0    NO
        CALL    WEEDSP    WEED MORE SPACES
        CPI     ','      COMMA FOLLOW?
        JNZ     ERR1      NO, ERROR
```

\* LOOK FOR # TO FOLLOW

```
MOTONO  CALL    WEEDSP    WEED SPACES
        SHLD    PNTRB     SAVE POINTER
        CPI     0600
        JC      ERR2      <0, NO GOOD
        CPI     0720
        JNC     MODFC      >9, NOT A #
```

\* FOUND A #, PROCESS IT

```
        CALL    OFFGEN    GENERATE OFFSET OCTAL #
        XCHG
        CALL    ILNTST    TEST FOR LEGAL #
        XCHG
        JZ      ERR2      NO GOOD
        MOV     A,B        GET CHAR COUNT IN (A)
        ANA     A          IS IT 0?
        JZ      ERR2      YES, NO # FOUND
        STA     CHRCNT     SAVE COUNT
        CALL    WEDSP      WEED SPACES
        STA     CFLAG      SAVE COMMA FLAG
```

\*\* STATUS REPORT:

\*\* <DE>=LINE NO. TO LOOK FOR

\*\* PNTRB=SET

\*\* CFLAG=SET

\*\*\* CALCULATE THE NEW LINE NO.

```
LHLD    INCREM
MOV     B,H              (BC)=INCREMENT
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

```

      MOV      C,L
      LHALD   LINNO      STARTING LN #
      PUSH    H          PUT IT ON THE STACK
      LXI     H,PROGRAM  POINT TO PROGRAM
GENCON  MOV     A,M       DOES LOOK # MATCH
      INX     H
      CMP     E
      JNE     GENC       NO MATCH
      MOV     A,M
      CMP     D
      JE      MATCH      YES, GOT IT
GENC    INX     H
      XCHG    (HL)=LOOK #, (DE)=PNTR
      CALL    FNDZRO      MOVE TO END OF LINE
      CALL    ENDTST      IS THIS THE END??
      DCX     D           BACK UP POINTERS
      DCX     D
      JZ      ERR3        YES, LINE NOT FOUND
      XCHG    (DE)=LOOK #, (HL)=PNTR
      XTHL    (HL)=LN #, (STACK)=PNTR
      DAD     B           ADD INCREMENT
      JC      ERR3        OVERFLO
      XTHL    (HL)=PNTR, (STACK)=NEW LN #
      JMP     GENCON      DO MORE

*      FOUND THE LINE

MATCH  POP      H         GET NEW LN #
      XCHG    SWAP
      MOV     A,H         ARE THEY EQUAL??
      CMP     D
      JNZ     GENASC      NO
      MOV     A,L
      CMP     E
      JZ      MCON        =, NO CHANGE NECESSARY

***    GENERATE ASCII STRING FROM VALUE
*      ENTER:  (DE)=VALUE
*      EXIT:   (DE)=ADDRESS OF LAST BYTE
*              (A) =LENGTH OF STRING DECODED
*      USES:   A,F,D,E,H,L

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

```
GENASC  CALL    IFLT      CHANGE # TO FLOATING PT.
        LXI     H,BUFFER  STORAGE FOR STRING
        CALL    FTA       CHANGE F.P. TO ASCII
        ANA     A
        JZ      ERR2      VALUE WAS 0
```

\* SUPPRESS LEADING ZEROS

```
        MOV     B,A        MOVE COUNT TO (B)
        LXI     H,BUFFER  POINT TO STRING
        INX     H          REMOVE LEADING AND
        DCR     B          TRAILING SPACES
        DCR     B
SUPPR   MOV     A,M        GET A CHAR
        CPI     '0'       ASCII 0?
        JNZ     CHECK     NO, DONE
        INX     H          YES, STEP AWAY
        DCR     B          COUNT-1
        JMP     SUPPR     DO MORE
```

\* STATUS REPORT:

```
*      (HL)=NEW BUFFER POINTER
*      (DE)=ADDRESS OF END OF BUFFER
*      (B) =# OF CHAR'S IN BUFFER
```

\* CHECK STRING PLACEMENT LENGTHS

\* (B)='NEW' STRING LENGTH

```
CHECK  PUSH     H          SAVE BUFFER POINTER
        LHLD    PNTRB     INTERNAL LINE POINTER
        LDA     CHRCNT    'OLD' STRING LENGTH
        CMP     B          COMPARE WITH 'NEW'
        JZ      XFER      EQUAL
        JP      CLOSE     'OLD'>'NEW'
```

\* NEW(B) > OLD(A) OPEN UP LINE

```
        MOV     C,A        SAVE 'OLD' IN (C)
        PUSH    H          SAVE (HL)
        LXI     H,COMMSTR  START OF COMMAND STRING
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

	CALL	CHARCNT	COUNT CHARACTERS
	ADD	B	GENERATE NEW
	SUB	C	COUNT
	POP	H	RESTORE (HL)
	CPI	100	MORE THAN 99 CHAR'S??
	JNC	ERR3	YES, TO MANY CHAR'S
	MVI	M,0	STORE STOP CHAR
	INX	H	
	XCHG		
	CALL	FNDZRO	FIND EOL ADDR.
	DCX	D	(DE)=FROM ADDR.
	MOV	A,B	NEW TO (A)
	SUB	C	SUBTRACT OLD
	ADD	E	'TO' ADDR IN (HL)
	MOV	L,A	
	MVI	A,0	
	ADC	D	
	MOV	H,A	
	LDAX	D	GET 1ST BYTE
	MOV	M,A	MOVE IT
OPEC	DCX	D	BUMP POINTERS
	DCX	H	
	LDAX	D	GET A BYTE
	MOV	M,A	MOVE IT
	ANA	A	DONE?
	JNZ	OPEC	NO, MOVE MORE
	JMP	XFER	YES, GO TRANSFER
*	OLD(A) > NEW(B) CLOSE UP		
*	(HL)='TO' ADDRESS		
CLOSE	SUB	B	'OLD' - 'NEW'
	ADD	L	'FROM' ADDR IN (DE)
	MOV	E,A	
	MVI	A,0	
	ADC	H	
	MOV	D,A	
CLOC	LDAX	D	GET A BYTE
	MOV	M,A	MOVE IT
	INX	D	BUMP POINTERS
	INX	H	

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

	ANA	A	DONE?
	JNZ	CLOC	NO, MOVE MORE
* TRANSFER BUFFER TO LINE			
XFER	LHLD	PNTRB	LOAD LINE POINTER
	XCHG		MOVE IT TO (DE)
	POP	H	RESTORE BUFFER POINTER
XFR	MOV	A,M	GET BUFFER CHAR
	STAX	D	MOVE IT
	INX	H	
	INX	D	
	DCR	B	DONE?
	JNZ	XFR	NO, MOVE MORE
*** THE STRING IS FIXED			
* GO BACK AND CHECK FOR MORE			
* LINE FIXING			
MCON	LHLD	PNTRB	GET POINTER
	LDA	CFLAG	GET COMMA FLAG
	CPI	','	WAS IT A COMMA?
	JNZ	MODFC	NO, CONTINUE
MCLOP	MOV	A,M	YES, GO BACK & FIND IT
	CPI	','	
	JZ	MOTONO	GOT IT, NEW POINTER SET
	INX	H	
	JMP	MCLOP	
EJECT			
***	PROCESS ERRORS		
ERR1	CALL	ERROR	SEND ERROR MESSAGE
	JMP	MODFC	CONTINUE
ERR4	POP	B	REMOVE A PUSH
ERR3	POP	B	REMOVE A PUSH
ERR2	CALL	ERROR	SEND ERROR MESSAGE
	JMP	MCON	TEST FOR TYPE OF CONTINUE
* LINE COMPLETED, CHECK FOR ROOM			

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

\* IN THE PROGRAM

MODFDN	LXI	H, COMMSTR	COMMAND STRING POINTER
	CALL	CHARCNT	COUNT 'NEW' CHAR'S
	MOV	B, A	'NEW' IN (B)
	LDA	OLD CNT	'OLD' IN (A)
	CMP	B	(<'OLD'-'NEW')
	JNC	MODOK	'NEW' <= 'OLD'
	MOV	C, A	'NEW' > 'OLD'
	MOV	A, B	
	SUB	C	GENERATE DIFF.
	LHLD	OLDEND	OLD MEMORY END
	ADD	L	GENERATE NEW END
	MOV	E, L	
	MVI	A, 0	
	ADC	H	
	MOV	D, H	
	LHLD	HIGHMEM	CONF'D HIGH MEMORY
	MOV	A, L	SUB NEW FROM LIMIT
	SUB	E	
	MOV	A, H	
	SBB	D	
	JC	MODNG	WON'T FIT

\* EXIT MODIFY ROUTINE

MODOK	XRA	A	EVERYTHINGS OK
	RET		

MODNG	CALL	ERROR	LINE WON'T FIT
	MVI	A, 1	
	ANA	A	
	RET		

STL 'SUBROUTINES'

EJECT

\* GENERATE OFFSET OCTAL VALUE

OFFGEN	LXI	D, 0	
	MOV	B, D	
OFFGE	CALL	GDIGIT	GET A DIGIT



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER - IGNATIUS < CONT'D >

RC		NOT A DIGIT-DONE
PUSH	H	SAVE POINTER
XCHG		
DAD	H	MULTIPLY (HL) BY 10
MOV	D,H	
MOV	E,L	
DAD	H	
DAD	H	
DAD	D	
JC	ERR4	OVERFLO
MOV	E,A	
MVI	D,0	
DAD	D	ADD NEXT DIGIT
JC	ERR4	OVERFLO
XCHG		
POP	H	RESTORE POINTER
INR	B	COUNT A DIGIT
INX	H	NEXT DIGIT
JMP	OFFGE	

\* SEND ERROR MESSAGE TO CONSOLE

ERROR	PUSH	H	SAVE
	RST	7	SEND MESSAGE
	DB	10,'ERR - LN# '	
	LHLD	CURLIN	GET LINE NO.
	XCHG		PLACE IN (DE)
	CALL	TDI	PRINT IT
	CALL	CRLF	
	LXI	H,ERRCNT	POINT TO ERROR COUNT
	INR	M	ADD 1
	POP	H	RESTORE
	RET		
END	ENTRY		

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS

```

      TITLE      'BASIC RENUMBER AND MERGE - VERSION 2.0'
* ****
* BASIC RENUMBER AND MERGE PROGRAM - VERSION 2.0
* WILLIAM MOSS - 10/21/78
* ****
*
* CALLS TO HEATH ROM AND CONSOLE DRIVER ROUTINES
*
DLV      EQU      000053A    DELAY ROUTINE
WNB      EQU      003024A    WRITE NEXT BYTE
WNP      EQU      003017A    WRITE NEXT PAIR
CRCSUM   EQU      040027A    CHECKSUM DATA
RNB      EQU      002331A    READ NEXT BYTE
SRS      EQU      002265A    SCAN FOR TAPE RECORD START
RNP      EQU      002325A    READ NEXT PAIR
TPERRX   EQU      040031A    FRONT PANEL TAPE CANCEL ROUTINE JUMP
ABUSS    EQU      040024A    ADDRESS FPLED STORAGE
$RCHAR   EQU      040144A    CONSOLE READ CHARACTER
$WCHAR   EQU      040147A    CONSOLE WRITE CHARACTER
$PRSCL   EQU      040152A    PRESET CONSOLE UART
COLNO    EQU      040253A    CONSOLE COLUMN NUMBER
*
*****
*
* COMMAND CODES IN EBHB
*
.GOSUB.   EQU      2170
.GOTO.    EQU      2200
.THEN.    EQU      2610
*
*****
*
* VARIABLE DEFINITIONS
*
IBUFLEN   EQU      6        LENGTH OF CONSOLE INPUT BUFFER
TITLEN    EQU      80       LENGTH OF TITLE BUFFER
*
*****
*
      ORG      040100A
      JMP      INIT
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

                JMP      COMMAND
*
*****
*
                ORG      040250A
                DW       040110A      JUMP TO RETURN ON CNTRL A THRU D
*
*****
*
                ORG      041144A      SET JUST ABOVE CONSOLE DRIVER NORMAL
LY (041144A)
                INIT     CALL      $PRSCl      SET CONSOLE UART
                CALL     PRINT
                DB       0,'BASIC RENUMBER PROGRAM - VERSION 2.0',0,200
Q
                COMMAND CALL      PRINT
                DB       0,'COMMAND (LOAD, DUMP, RENUMBER, MERGE, TITLE
)?',2400
                CALL     QUERY
                DB       'LDRM', 'T'+2000
                ANA      A
                JZ       COMJMP      IF NOT NOW LOADING, SEE IF PREVIOUS
LOAD DONE
                LHL      TXEND      HL=0 IF NO LOAD HAS BEEN COMPLETED
                DCR      H
                JM       LOF
                COMJMP   CALL      JMPTBL
                DW       XLOAD,XDUMP,RENUM,MERGE,XTITLE
*
****
*
                LOF      CALL      PRINT
                DB       'NO PROGRAM LOADED YET',2000
                JMP      COMMAND
*
****
*
                XDUMP    LHL      TXEND
                XCHG

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

XLOAD      LXI      H, TXSTART
           CALL     LODUMP
           JMP      COMMAND

*
***
*
RENUM      CALL     PRINT
           DB        0, 'RENUMBERING TO START A', 'T'+2000
           CALL     INPNUM      HL=INPUT INTEGER
           SHLD     LNSTART
           SHLD     NEWLN
           CALL     PRINT
           DB        'LINE NUMBER INCREMEN', 'T'+2000
           CALL     INPNUM
           SHLD     LNINC
           LHD      TXEND
           MOV      D, H
           MOV      E, L
           INR      H
           INR      H      SKIP 2 PAGES TO ALLOW FOR EXPANSION
           SHLD     LNTABLE
           DCX      H
           MOV      B, H
           MOV      C, L      BC="TEMPORARY END OF TEXT"
           LXI      H, TXSTART
           CALL     SHIFTER
           LHD      LNTABLE
           XCHG
           SHLD     TEMPST
LNFOUND    MOV      A, M
           MOV      B, A      B=LOW BYTE OF OLD LINE #
           STAX     D
           INX      H
           INX      D
           MOV      A, M
           STAX     D      STORE HIGH BYTE
           ANA      B
           CPI      3770
           JZ       TBLDONE   IF LINE # TABLE COMPLETED

           PUSH     D

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

	XCHG		
	LHLD	NEWLN	REPLACE OLD # WITH NEW
	XCHG		
	DCX	H	
	MOV	M,E	
	INX	H	
	MOV	M,D	
	POP	D	
	INX	H	
	INX	D	
	CALL	INCLN	
FINDZER	MOV	A,M	
	INX	H	
	ANA	A	
	JNZ	FINDZER	
	JMP	LNFOUND	
***			
SHIFTER	PUSH	B	MOVES BLOCK UP IN MEMORY
	PUSH	D	
	CALL	DMH1	
	MOV	C,E	
	MOV	B,D	BC=COUNT
	POP	H	HL=OLD END OF BLOCK
	POP	D	DE=NEW END OF BLOCK
MOUBYTE	MOV	A,M	
	STAX	D	
	DCX	H	
	DCX	D	
	DCX	B	
	MOV	A,B	
	ORA	C	
	JNZ	MOUBYTE	
	INX	D	
	RET		
***			
TBLDONE	LXI	H, TXSTART	
	DCX	H	
	XCHG		DE-> NEW TEXT
	LHLD	TEMPST	HL-> OLD TEXT

<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```
NEXTLIN  MOV      A,M
          INX      D
          STAX     D
          MOV      C,A
          INX      H
          MOV      A,M
          INX      D
          STAX     D
          PUSH     H
          MOV      H,A
          MOV      L,C
          SHLD     TEMPDAT  SAVE CURRENT LINE NUMBER
          POP      H
          ANA      C
          CPI      377Q
          JZ       FINISH   IF END OF TEXT

          INX      H
SEEKLN    MOV      A,M
          INX      D
          STAX     D
          CPI      .GOSUB.
          JZ       LNUMB

          CPI      .GOTO.
          JZ       LNUMB

          CPI      .THEN.
          JZ       LNUMB

          INX      H
          ANA      A
          JNZ      SEEKLN

          JMP      NEXTLIN

***
LNUMB     INX      H          FOUND A JUMP TO LINE #
          MOV      A,M
          CPI      / /
          JZ       LNUMB     SKIP LEADING BLANKS
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

	CALL	TSDIG	
	JC	SEEKLN+1	IF NO NUMBER FOLLOWS
	PUSH	D	STACK= NEW TEXT->
	CALL	ASCDE	DE=BINARY VALUE OF OLD LINE NUMBER
	XTHL		STACK=OLD TEXT->, HL=NEW TEXT->
	PUSH	H	STACK=NEW TEXT->
	LHLD	LNSTART	
	SHLD	NEWLN	
LNTEST	LHLD	LNTABLE	
	MOV	C,M	
	INX	H	
	MOV	B,M	
	INX	H	
	MOV	A,C	
	CMP	E	
	JNZ	NOMATCH	
	MOV	A,B	
	CMP	D	
	JNZ	NOMATCH	
	LHLD	NEWLN	FOUND A MATCH
	XCHG		DE=NEW LINE #
	POP	H	HL-> NEW TEXT
	INX	H	
	CALL	DEASC	PUT NEW LINE # IN ASCII AT HL
	XCHG		
	POP	H	HL-> OLD TEXT, DE-> NEW TEXT
EW TEXT	DCX	D	DE MUST POINT TO LAST CHARACTER IN N
COMCHK	MOV	A,M	CHECK FOR ANOTHER NUMBER AFTER A COM
MA			
	CPI	','	
	JNZ	SEEKLN	IF COMMA, LOOK FOR ANOTHER LINE #
	INX	D	
	STAX	D	WRITE COMMA IN NEW TEXT
	JMP	LNUMB	
***			
FINISH	INX	D	

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

                                HL=NEW TXEND
                                M,0
                                SHLD TXEND
                                CALL PRINT
                                DB 'RENUMBERING COMPLETED',2000
                                JMP COMMAND

***
NOMATCH  MOV A,C
          ANA B
          CPI 3770
          JZ  NEVER      IF CAN'T MATCH AT ALL

          CALL INCLN
          JMP LNTEST

***
NEVER    CALL PRINT
          DB 7,'REFERENCE TO NONEXISTENT LINE NUMBER:',2400
          CALL WRITED E  WRITE BINARY IN DE AS DECIMAL ON CON

SOLE
          CALL PRINT
          DB ' IN LINE NUMBER',2400
          PUSH D
          LHLD TMPDAT    HL= CURRENT LINE NUMBER
          XCHG
          CALL WRITED E
          POP D
          CALL CRLF
          POP H          HL-> NEW TEXT
          INX H
          MVI M,'['     BRACKET ERRONEOUS LINE # IN NEW TEXT
          INX H
          CALL DEASC
          MVI M,']'
          XCHG           DE-> NEW TEXT
          POP H          HL-> OLD TEXT
          JMP COMCHK

*
***
*
WRITED E  PUSH D
          LXI H,INPBUF  USE INPBUF FOR TEMPORARY STORAGE

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

PUSH      H
CALL      DEASC
XCHG
POP       H          HL=INPBUF
CALL      DMH0       DE= DE-HL
MOV       A,E        A=LENGTH OF ASCII STRING FOR NUMBER
CALL      WRSTR
POP       D
RET

```

\*  
\*\*\*  
\*

```

INCLN     PUSH      H
          PUSH      D
          LHL D      NEWLN
          XCHG
          LHL D      LNINC
          DAD       D
          SHLD      NEWLN
          POP       D
          POP       H
          RNC

```

RETURN IF NO 16-BIT OVERFLOW

```

          CALL      PRINT
          DB        0,7,'NEW LINE NUMBER > 65535!  RELOAD BASIC TE
          XT AND TRY AGAIN.',7,2000
          LXI       H,TXEND+1
          MVI       M,0      FLAG THAT PROGRAM MUST BE RELOADED
          POP       PSW      CLEAR STACK
          JMP       COMMAND

```

\*  
\*\*\*  
\*

```

INPNUM     LXI       H,INPBUF
          PUSH      H
          MVI       E,5      LIMIT INPUT TO 5 CHARACTERS
          ERMI      IBUFLEN-6 INPUT BUFFER MUST BE AT LEAST 6 BYTE

          CALL      INPSTR
          MVI       M,'X'    MARK END OF NUMBER WITH A NONDIGIT
          POP       H

```

S

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

                                CMP      E      A= # BYTES INPUT, E= 5
                                JC       NUMOK   OK IF <5 BYTES

                                MVI      A,'6'    ELSE CHECK 1ST DIGIT
                                CMP      M
                                JC       OUTFIT   NO GOOD IF DIGIT >'6'

NUMOK    PUSH      PSW      NZ= OK, Z= MUST CHECK FOR OVERFLOW
          CALL     ASCDE    DE= BINARY VALUE
          POP      PSW      RESTORE FLAGS
          JNZ      CHECK0

FLOW     MOV      H,D      1ST DIGIT IS A '6' SO CHECK FOR OVER

          MOV      L,E
          LXI      B,-5000  IF OVERFLOW THEN HL= FROM 0 TO 4465
          DAD      B
          JNC      OUTFIT

CHECK0   XCHG
          MOV      A,H
          ORA      L
          RNZ

OUTFIT   CALL     PRINT
          DB       7,'INPUT MUST BE AN INTEGER: 1 <= N <= 65535',
0,'REENTER I',T'+2000
          JMP      INPNUM

*
***
*
INPSTR   PUSH      B
          CALL     PRINT
          DB       '?',2400
          MVI      C,0
NEXTCHR  CALL     $RCHAR
          CPI      150     'CR'
          JZ       STRDONE

          CPI      177Q    RUBOUT (BACKSPACE)
          JZ       BS

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

	CPI	/ /	IGNORE NON-PRINTED CHARACTERS
	JC	BELL	
	MOV	B,A	B=CHAR
	MOV	A,C	
	CMP	E	IS STRING TOO LONG?
	JZ	BELL	
CHROK	MOV	M,B	
	INR	C	
	INX	H	
	MOV	A,B	
WC	CALL	WRCHAR	
	JMP	NEXTCHR	
***			
BS	DCR	C	
	JM	TOOFAR	
	CALL	PRINT	
	DB	100,400,2100	
	DCX	H	
	JMP	NEXTCHR	
***			
TOOFAR	INR	C	BACKSPACED TOO FAR
BELL	MVI	A,7	
	JMP	WC	
***			
STRDONE	MOV	A,C	
	POP	B	
	CALL	CRLF	
	RET		
*			
***			
*			
TSDIG	CPI	'0'	SEE IF A IS A ASCII NUMERIC
	RC		
	CPI	'9'+1	
	CMC		
	RET		

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

***
ASCDE   LXI      D,0      ASCII IN HL-> BINARY IN DE
        MOV      A,M
DIGIT   CALL     TSDIG
        RC

        PUSH     H
        MOV      L,E
        MOV      H,D
        DAD      H        X2
        DAD      H        X4
        DAD      D        X5
        DAD      H        X10
        SUI      /0/
        MOV      E,A
        MVI      D,0
        DAD      D
        XCHG
        POP      H        DE= 10 * DE + DIGIT
IGBLK   INX      H
        MOV      A,M
        CPI      / /
        JZ       IGBLK    IGNORE BLANKS

        JMP      DIGIT

*
***
*
DEASC   MOV      A,D      BINARY INTEGER IN DE TO ASCII AT HL
        ORA      E
        JZ       ITIS0

        PUSH     H
        XCHG
        CALL     BCD      10'S & 1'S
        MOV      C,A
        CALL     BCD      1000'S & 100'S
        MOV      D,L      10000'S
        MOV      B,A
        MVI      E,0      SUPPRESS ZEROES
        POP      H

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

	CALL	DIG2	HIGH TWO
	MOV	D,B	
	CALL	DIG2	MID TWO
	MOV	D,C	
DIG2	MOV	A,D	PUT OUT TWO DIGITS IN D
	RRC		
	RRC		
	RRC		
	RRC		
	CALL	PUT	FIRST LEFT ONE
	MOV	A,D	
PUT	ANI	15	PUT OUT RIGHT DIGIT IN 'A'
	CMP	E	LEAD 0?
	RZ		IF SO, DON'T
ITIS0	MVI	E,48	PRINT ZEROES
	ADD	E	ASCII OFFSET
	MOV	M,A	
	INX	H	
	RET		

\*\*\*

\*

\* CONVERT TWO DECADES

\*

BCD	MVI	D,16	BIT COUNT
	XRA	A	START AT 0
CUT	DAD	H	SHIFT LEFT
	ADC	A	A=2*A+CARRY
	DAA		DO IN DECIMAL
	JNC	TST	

	INX	H	BACK INTO HL
TST	DCR	D	
	JNZ	CUT	
	RET		

\*

\*\*\*

\*

\* MERGE ROUTINE TO COMBINE TWO BASIC PROGRAMS

\*

MERGE	XRA	A	A=0 SIGNIFIES 'LOAD' TO LODUMP ROUTI
NE			

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS

	LHLD	TXEND	
	SHLD	START2	SAVE OLD PROG1 END TEMPORARILY
	INX	H	
	CALL	LODUMP	LOAD PROG2
	ANA	A	A=0 IF BAD LOAD, A=1 IF GOOD LOAD
	JZ	COMMAND	EXIT IF THERE IS TAPE ERROR
	LXI	H,0	
	DAD	SP	
	DCR	H	HL=1 PAGE UNDER STACK
	SHLD	END2	=END OF PROG2
	MOV	B,H	
	MOV	C,L	BC->DESTINATION OF BLOCK SHIFT
	LHLD	TXEND	HL->END OF PROG2
	XCHG		DE-> "
	LXI	H, TXSTART	
	CALL	SHIFTER	
	SHLD	TXEND	HL= TXSTART-1
	XCHG		
	SHLD	START1	=NEW START OF PROG1
	XCHG		
	LXI	H, TXSTART	
IFT	CALL	DMH1	DE= START1-TXSTART+1 = # BYTES OF SH
	LHLD	START2	HL= OLD PROGRAM END
	DAD	D	HL= NEW START OF PROG2
SORTIT	SHLD	START2	
	LHLD	START2	
	MOV	C,M	
	INX	H	
	MOV	B,M	BC= CURRENT LINE # OF PROG2
	LHLD	START1	
	INX	H	
	MOV	A,M	
	DCX	H	
	SUB	B	SUBTRACT CURRENT LINE # OF PROG1
	JNZ	FLAGSET	
	MOV	A,M	
	SUB	C	
	JZ	EQUAL	

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

FLAGSET	JNC	USE2	
	CALL	MOULIN	
	SHLD	START1	
	JMP	SORTIT	
***			
EQUAL	INX	H	SKIP LINE IN PROG1 IF EQUAL LINE #'S
	INX	H	
EQ1	MOV	A,M	
	INX	H	
	ANA	A	
	JNZ	EQ1	
USE2	SHLD	START1	
	LHLD	START2	
	CALL	MOULIN	
	SHLD	START2	
	XCHG		
	LHLD	END2	
	CALL	DMH0	SUBTRACT END2 FROM START2
	JC	SORTIT	IF MERGE NOT COMPLETE
	JMP	COMMAND	
***			
MOULIN	XCHG		MOVES LINE FROM HL TO TXEND+1
	LHLD	TXEND	
	MVI	C,2	
ML1	INX	H	
	LDAX	D	
	MOV	M,A	
	INX	D	
	DCR	C	
	JP	ML1	SKIP LINE # (1ST TWO BYTES OF LINE)
	ANA	A	
	JNZ	ML1	GO UNTIL ZERO ENCOUNTERED
	SHLD	TXEND	
	XCHG		
	RET		

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

\*  
\*\*\*  
\*

XTITLE	CALL	PRINT
	DB	'CURRENT TITLE:',2400
	LXI	H,TITLE
	MOV	A,M
	PUSH	H
	INX	H
	PUSH	H
	CALL	WRSTR
	CALL	PRINT
	DB	0,'NEW TITL', 'E'+2000
	POP	H
	MVI	E,TITLEN
	CALL	INPSTR
	POP	H
	MOV	M,A
	JMP	COMMAND

\*  
\*\*\*  
\*

QUERY	XTHL		HL=1ST ADDR OF LIST
	PUSH	D	
Q1	MVI	D,0	
	CALL	\$RCHAR	
	PUSH	H	SAVE ADDR OF LIST START
	MOV	E,A	SAVE CHARACTER INPUT IN E
CKLIST	MOV	A,M	
	ANI	1770	MASK BIT 7
	CMP	E	
	JZ	MATCH	IF LIST MATCHES INPUT CHARACTER
	MOV	A,M	
	INR	D	
	INX	H	
	RAL		SEE IF BIT 7 IS SET
	JNC	CKLIST	
	CALL	PRINT	LIST EXHAUSTED WITHOUT MATCH
	DB	2070	BELL



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

        POP        H
        JMP        Q1
***
MATCH   MOV        A,E
        CALL       $WCHAR      ECHO INPUT CHARACTER
        CALL       CRLF
FINDEND MOV        A,M          FIND END OF LIST TO SET RETURN
        RAL
        INX        H
        JNC        FINDEND

        MOV        A,D
        POP        D           CLEAR STACK
        POP        D
        XTHL
        RET
*
***
*
JMPTBL  XTHL
        PUSH       PSW
        ADD        A          A=A*2
        CALL       ADAHL
        MOV        A,M
        INX        H
        MOV        H,M
        MOV        L,A
        POP        PSW
        XTHL
        RET
*
***
*
ADAH    PUSH       D           ADDS A TO HL
        MOV        E,A
        MVI        D,0
        DAD        D
        POP        D
        RET
*
***

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

*
PRINT      XTHL      CONSOLE STRING PRINTER (STRING FOLLO
WS CALL)

    PRNT1      PUSH      PSW
                MOV      A,M
                INX      H
                CALL     WRCHAR
                ANA      A
                JP       PRNT1      IF MORE TO COME

                POP      PSW
                XTHL
                RET

*
***
*
WRCHAR      PUSH      PSW      WRITES A CHARACTER TO CONSOLE
                ANI      177Q
                ANA      A
                CZ       CRLF      @=CR/LF CHARACTER
                CALL     $WRCHAR
                PUSH     H
                LXI      H,COLNO
                INR      M
                CPI      40Q      'SPACE'
                JNC      **+4

                DCR      M      NONPRINTED CHARACTER
                CPI      10Q      'BACKSPACE'
                JNZ      **+4

                DCR      M
                CPI      15Q      'CR'
                JNZ      **+5

                MVI      M,@      NEW LINE; SET CURSOR TO @
                MOV      A,M
                INX      H      HL=CONSOLE MAX WIDTH ADDR
                CMP      M
                CNC      CRLF      CR/LF IF LINE TOO LONG
                POP      H

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

                POP      PSW
                RET

*
***
*
CRLF           CALL     PRINT
                DB       150,212Q   CR & LF
                RET

*
***
*
LODUMP         SHLD     TEMPDAT      LOAD/DUMP ROUTINE TO MAGNETIC TAPE
                LXI      H,TPXIT
                SHLD     TPERRX
                LXI      H,0
                DAD      SP
EXIT           SHLD     TPXIT+1      SAVE CURRENT STACK POINTER FOR ERROR

                CALL     PRINT
                DB       'TURN ON THE TAPE RECORDER FOR TAPE',240Q
                ANA      A           IF A=0 THEN LOAD, ELSE DUMP
                JNZ      DUMP

                CALL     PRINT      LOAD ROUTINE
                DB       'LOADING',200Q
                LXI      H,-2*256    COMPLEMENT OF 002.000
                CALL     LODREC
                LXI      H,TITLE
                MOV      M,E         STORE LENGTH OF TITLE
                INX      H
                CALL     LOA1       LOAD TITLE
                CALL     PRINT
                DB       'FOUND',240Q
                LXI      H,TITLE
                MOV      A,M
                INX      H
                CALL     WRSTR
                CALL     CRLF
                LXI      H,1000Q-202Q*256-1  COMPLEMENT OF 202.001
                CALL     LODREC
                LHLD     TEMPDAT

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

	CALL	LOA1	LOAD BASIC TEXT
	DCX	H	
	SHLD	TXEND	SAVE ADDRESS OF END OF BASIC TEXT
	MVI	A,1	A=1 MEANS GOOD LOAD
	RET		
***			
LOA1	CALL	RNB	
	MOV	M,A	
	INX	H	
	XCHG		
	SHLD	ABUSS	DISPLAY COUNT ON FPLED5
	XCHG		
	DCX	D	
	MOV	A,D	
	ORA	E	
	JNZ	LOA1	
	XCHG		DE=LAST MEMORY ADDRESS + 1
	CALL	RNP	CHECK CRC SUM
	LHLD	CRC SUM	
	MOV	A,H	
	ORA	L	
	XCHG		
	JZ	TOT	IF CRC OK, THEN TURN OFF TAPE
	CALL	PRINT	
	DB	'CHECKSUM',2400	
TPXIT	LXI	SP,0	SET RETURN TO CALL OF LODUMP ROUTINE
	CALL	PRINT	
	DB	'ERROR!',7,2000	
	JMP	TOT	
*			
***			
*			
WRSTR	ANA	A	PRINTS A BYTES STARTING AT HL
	RZ		
	PUSH	PSW	
	MOV	A,M	
	INX	H	
	CALL	WRCHAR	



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

      POP      PSW
      DCR      A
      JMP      WRSTR

*
***
*
LODREC  PUSH      H          STACK=COMP OF EXPECTED TYPE.RECORD
        CALL      SRS
        MOV       L,A
        XTHL
        DAD       D          HL=COMP, STACK=COUNT
        MOV       A,H        DE=TYPE.RECORD
        ORA       L
        POP       D          DE= BYTE COUNT
        RZ          RETURN IF NO TYPE OR SEQUENCE ERROR

        CALL      PRINT
        DB        'TYPE OR SEQUENCE',2400
        JMP      TPXIT

*
***
*
DUMP    PUSH      D
        CALL      PRINT
        DB        'DUMPING',2000
        LXI       H,TITLE
        MOV       E,M
        MVI       D,0        DE=BYTE COUNT OF TITLE
        INX       H          HL=TITLE ADDR
        LXI       B,2*256    BC=TYPE.RECORD
        CALL      WRREC
        LHLD      TEMPDAT    HL=START ADDR FOR DUMP
        POP       D          DE=END ADDR FOR DUMP
        CALL      DMH1
        LXI       B,202001A  BC=TYPE.RECORD
        CALL      WRREC
        RET

*
***
*
DMH1    INX       D          ROUTINE: DE = (DE-HL)+1

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: RENUMBER/MERGE - MOSS < CONT'D >

```

DMH0      MOV      A,E      ROUTINE: DE= DE-HL
          SUB      L
          MOV      E,A
          MOV      A,D
          SBB      H
          MOV      D,A
          RET

*
***
*
WRR0      PUSH     H
          MVI      A,1
          OUT      3710      SET UP TAPE CONTROL
          MVI      A,260     'SYN'
          MVI      L,30
WRR1      CALL     WNB        WRITE 30 SYN CHARACTERS
          DCR      L
          JNZ      WRR1

          MVI      A,2        'STX'
          CALL     WNB
          MOV      H,L        HL=0
          SHLD     CRCSUM     CLEAR CRCSUM
          MOV      H,B        H=TYPE
          MOV      L,C        L=SEQUENCE #
          CALL     WNP
          POP      H          HL=START ADDR
          XCHG
          CALL     WNP        HL=COUNT TO BE WRITTEN
          XCHG
WRR2      MOV      A,M
          CALL     WNB
          XCHG
          SHLD     ABUSS      DISPLAY COUNT ON FRONT PANEL
          XCHG
          INX      H
          DCX      D
          MOV      A,D
          ORA      E
          JNZ      WRR2

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE

```

      TITLE 'ASSEMBLY LANGUAGE UTILITY ROUTINES'
      STL 'ASSEMBLY CONTROL CONSTANTS'
*      GENERAL H8 ASSEMBLY LANGUAGE UTILITY ROUTINES
*      WRITTEN: R.T. PRICE
*      DATE: AUGUST 1978
*      FEATURES:
*      ENTRY THRU STANDARD UTILITY ENTRY POINTS
*      SELECTIVE ASSEMBLY
*      MAY BE ASSEMBLED SEPARATELY FROM CALLING (MAIN) R
OUTINE
*      RUN ALONE (ORG 40100A) OR WITH HBUG (ORG 55000A)
*
*      FUNCTIONS:
*      CONSOLE DRIVER $RCHAR,$WCHAR,$PRSC
*      CONSOLE OUTPUT .TYPE,.TOP
*      STRING COMPARISON .SCAN,.CMPAR
*      BLOCK MOVE .MOUCL,.MOUDB,.FILL
*      TABLE LOOKUP .LOKLE,.LOKE,.LOKGE,.
LOKL
*      DEBUGGING SUPPORT .ERMES,.TRACE
*      TAPE INPUT/ .LOPEN,.LREAD,.LCLOSE
*      OUTPUT .DOPEN,.DWRITE,.DCLOS
E
*      INTEGER ARITHMETIC .ADD4,.SUB4,.LOAD4,.C
LEAR4,
*      .STORE4,.NEG4,.NEG4U
*      ASCII/BINARY CONVERSION .CONA4,.CON4A,.SFTL4
*
      ORG 40100A
*
*UTILITY ROUTINES
*
*      SELECTIVE ASSEMBLY CONTROL CONSTANTS---
*      PROVIDE SELECTIVE CONTROL OF ROUTINES IN PACKAGE
WHICH ARE ASSEMBLED
*      (=1 -- ROUTINE ASSEMBLED)
*      (=0 -- ROUTINES NOT ASSEMBLED UNLESS
REQUIRED BY ANOTHER ROUTINES)
*
*UTILTY EQU *
A.ARIT4 EQU 1
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE < CONT'D >

```
A.CONA4 EQU 1
A.CON4A EQU 1
IF .UTILITY-55000A
A.CONSL EQU 0
ELSE
A.CONSL EQU 1
ENDIF
A.CMPAR EQU 1
A.DOPEN EQU 1
A.DWRIT EQU 1
A.DCLOS EQU 1
A.DTAPE EQU 1
A.FILL EQU 1
A.LOKUP EQU 1
A.LCLOS EQU 1
A.LOPEN EQU 1
A.LREAD EQU 1
A.LTAPE EQU 1
A.MOVD8 EQU 1
A.MOVEL EQU 1
A.SCAN EQU 1
A.SFTL4 EQU 1
A.TYPE EQU 1
A.TRACE EQU 1
```

```
*
/ STL 'LTAPE AND DTAPE PARAMETERS, UTILITY ENTRY POINTS
```

EJECT

```
* NOTE--LTAPE/DTAPE PARAMETERS MUST BE PRESENT TO INSURE C
ORRECT POSITIONING
```

```
* OF ENTRY POINTS - FORMAT CAN NOT BE CHANGED
WITHOUT ALTERING CALLS IN MAIN
```

TITLE ' ASSEMBLY LANGUAGE UTILITY ROUTINES'

STL 'ASSEMBLY CONTROL CONSTANTS'

```
* GENERAL H8 ASSEMBLY LANGUAGE UTILITY ROUTINES
```

```
* WRITTEN: R.T. PRICE
```

```
* DATE: AUGUST 1978
```

```
* FEATURES:
```

```
* ENTRY THRU STANDARD UTILITY ENTRY POINTS
```

```
* SELECTIVE ASSEMBLY
```

```
* MAY BE ASSEMBLED SEPARATELY FROM CALLING <MAIN> R
```

OUTLINE

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE

```

*          RUN ALONE (ORG 40100A) OR WITH HBUG (ORG 55000A)
*
*      FUNCTIONS:
*          CONSOLE DRIVER          $RCHAR,$WCHAR,$PRSC
*          CONSOLE OUTPUT          .TYPE,.TOP
*          STRING COMPARISON       .SCAN,.CMPAR
*          BLOCK MOVE              .MOUDB,.MOUDB,.FILL
*          TABLE LOOKUP           .LOKLE,.LOKE,.LOKGE,.
LOKL
*          DEBUGGING SUPPORT       .ERMES,.TRACE
*          TAPE INPUT/             .LOPEN,.LREAD,.LCLOSE
*          OUTPUT                  .DOPEN,.DWRITE,.DCLOS
E
*          INTEGER ARITHMETIC     .ADD4,.SUB4,.LOAD4,.C
LEAR4,
*          ASCII/BINARY CONVERSION .STORE4,.NEG4,.NEG4U
*          .CONA4,.CON4A,.SFTL4
*
*          ORG      40100A
*
*      UTILITY ROUTINES
*
*          SELECTIVE ASSEMBLY CONTROL CONSTANTS---
*          PROVIDE SELECTIVE CONTROL OF ROUTINES IN PACKAGE
WHICH ARE ASSEMBLED
*          (=1 -- ROUTINE ASSEMBLED)
*          (=0 -- ROUTINES NOT ASSEMBLED UNLESS
REQUIRED BY ANOTHER ROUTINES)
*
*      .UTILTY EQU      *
A.ARIT4 EQU      1
A.CONA4 EQU      1
A.CON4A EQU      1
IF      .UTILTY-55000A
A.CONSL EQU      0
ELSE
A.CONSL EQU      1
ENDIF
A.CMPAR EQU      1
A.DOPEN EQU      1
A.DWRIT EQU      1

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE < CONT'D >

A.DCLOS EQU 1  
A.DTAPE EQU 1  
A.FILL EQU 1  
A.LOKUP EQU 1  
A.LCLOS EQU 1  
A.LOPEN EQU 1  
A.LREAD EQU 1  
A.LTAPE EQU 1  
A.MOVD8 EQU 1  
A.MOUBL EQU 1  
A.SCAN EQU 1  
A.SFTL4 EQU 1  
A.TYPE EQU 1  
A.TRACE EQU 1

\*

STL 'LTAPE AND DTAPE PARAMETERS, UTILITY ENTRY POINTS

/

EJECT

\* NOTE--LTAPE/DTAPE PARAMETERS MUST BE PRESENT TO INSURE C  
ORRECT POSITIONING

\* OF ENTRY POINTS - FORMAT CAN NOT BE CHANGED  
WITHOUT ALTERING CALLS IN MAIN

\*

\*

\*

.LTAPE PARAMTERS

\*

.LOPENF DW 200 INPUT FILE LABEL CHARACTER MATCH COU  
NT (<=20)

.LOPENT DB 200 FILE LABEL  
COMPRESSED TEXT (003) OR BLOCKED VAR

TABLE LENGTH FILES (011)

.LOPENR DW 0A INPUT RECORD AREA -- INCLUDING LENGT  
H IN FIRST 2 BYTES

.LOPENE DW 0A ADDRESS OF END OF FILE ROUTINE

.LOPENW DW 0A ADDRESS OF INPUT WORK AREA -- INCLUD

ING LENGTH IN FIRST 2 BYTES

\*

.DTAPE PARAMETERS

\*

.DOPENF DW 600 OUTPUT FILE LABEL CHARACTER COUNT (<  
=60)

D5 600 FILE LABEL

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE < CONT'D >

.DOPENR	DW	0A	OUTPUT RECORD ADDRESS -- INCLUDING L
ENGTH IN FIRST 2 BYTES			
.DOPENT	DB	0110	COMPRESSED TEXT (003) OR BLOCKED VAR
IABLE LENGTH (011) FILES			
.DOPENW	DW	0A	ADDRESS OF OUTPUT WORK AREA
.DOPENL	DW	0A	MAX LENGTH OF OUTPUT BLOCK

\*

\* NOTE LTAPE/DTAPE PARAMETERS AND UTILITY ENTRY POINTS MUST MATCH CALLING SEQUENCE IN

\* MAIN PROGRAM EXACTLY

\*

JMP	\$RCHAR	
JMP	\$WCHAR	+3
JMP	\$PRSC	+6
JMP	\$CSINT	+9
JMP	.CMPAR	+12
JMP	.ERMES	+15
JMP	.FILL	+18
JMP	.LOKLE	+21
JMP	.LOKE	+24
JMP	.LOKGE	+27
JMP	.LOKL	+30
JMP	.MOUCL	+33
JMP	.MOUDB	+36
JMP	.SCAN	+39
JMP	.TYPE	+42
JMP	.TOP	+45
JMP	.DOPEN	+48
JMP	.DWRITE	+51
JMP	.DWRITE	+54
JMP	.DCLOSE	+57
JMP	.LOPEN	+60
JMP	.LREAD	+63
JMP	.LCLOSE	+66
JMP	.ADD4	+69
JMP	.SUB4	+72
JMP	.LOAD4	+75
JMP	.STORE4	+78
JMP	.CLEAR4	+81
JMP	.NEG4	+84
JMP	.NEG4U	+87



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE < CONT'D >

```

        JMP      .SFTL4          +90
        JMP      .CONA4          +93
        JMP      .CON4A          +96
        JMP      .TRACE          +99
        IF      A.CONSL+A.DTAPE+A.LTAPE
*      PAM8 CONSTANTS NOT ASSEMBLED
        ELSE
        STL      'PAM8 CONSTANTS'
        EJECT

*
*      CONSOLE DRIVER ROUTINE
*
BELL      EQU      0070
.UIVEC    EQU      040037A
*      USART BIT DEFINITIONS
UMI.1B    EQU      01000000B    1 STOP BIT
* UMI.HB    EQU      10000000B    1 1/2 STOP BITS
* UMI.2B    EQU      11000000B    2 STOP BITS
* UMI.PE    EQU      00100000B    EVEN PARITY
* UMI.PA    EQU      00010000B    USE PARITY
* UMI.L5    EQU      00000000B    5 BIT CHARACTERS
* UMI.L6    EQU      00000100B    6 BIT CHARACTERS
* UMI.L7    EQU      00001000B    7 BIT CHARACTERS
UMI.L8    EQU      00001100B    8 BIT CHARACTERS
* UMI.1X    EQU      00000001B    CLOCK X 1
UMI.16X    EQU      00000010B    CLOCK X 16
* UMI.64X    EQU      00000011B    CLOCK X 64
*      COMMAND INSTRUCTION BITS
UCI.IR    EQU      01000000B    INTERNAL RESET
UMI.RO    EQU      00100000B    READER ON CONTROL FLAG
UCI.ER    EQU      00010000B    ERROR RESET
UCI.RE    EQU      00000100B    RECEIVE ENABLE
UCI.IE    EQU      00000010B    ENABLE INTERRUPTS FLAG
UCI.TE    EQU      00000001B    TRANSMIT ENABLE
*      STATUS READ COMMAND BITS
* USR.FE    EQU      00100000B    FRAMING ERROR
* USR.OE    EQU      00010000B    OVERRUN ERROR
* USR.PE    EQU      00001000B    PARITY ERROR
USR.TXE    EQU      00000100B    TRANSMITTER EMPTY
USR.RXR    EQU      00000010B    RECEIVER READY
USR.TXR    EQU      00000001B    TRANSMITTER READY

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE < CONT'D >

```
*
*
*      I.O PORT ADDRESSES
*
IP.CDP EQU 3720  CONSOLE DATA IN PORT
OP.CDP EQU 3720  CONSOLE DATA OUT PORT
IP.CIS EQU 3730  CONSOLE INPUT STATUS IN PORT
OP.CIS EQU 3730  CONSOLE INPUT STATUS OUT PORT
IP.COS EQU 3730  CONSOLE OUTPUT STATUS IN PORT
OP.COS EQU 3730  CONSOLE OUTPUT STATUS OUT PORT
*
OP.TDP EQU 3700  TAPE DATA OUT PORT
IP.TDP EQU 3700  TAPE DATA OUT PORT
IP.TSP EQU 3710  TAPE STATUS IN PORT
OP.TSP EQU 3710  TAPE STATUS OUT PORT
*
$CSLCTL CONSOLE CONTROL FLAG BITS
CC.HLD EQU 01    SUSPEND OUTPUT
CC.DMP EQU 02    DISCARD OUTPUT
CC.CTLA EQU 0100 CTL-A
CC.CTLB EQU 0200 CTL-B
CC.CTLC EQU 0400 CTL-C
CC.CTLD EQU 1000 CTL-D
*
*      PORT ROUTINES
*
$CDIN IN IP.CDP CONSOLE DATA IN
$RET RET
$CDOUT OUT OP.CDP CONSOLE DATA OUT
RET
$CISI IN IP.CIS CONSOLE INPUT STATUS IN
RET
$CISO OUT OP.CIS CONSOLE INPUT STATUS OUT
RET
$COSI IN IP.COS CONSOLE OUTPUT STATUS IN
RET
$COSO OUT OP.COS CONSOLE OUTPUT STATUS OUT
RET
$TDIN IN IP.TDP TAPE DATA IN
RET
$TDOUT OUT OP.TDP TAPE DATA OUT
RET
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITY ROUTINES - PRICE < CONT'D >

```
$TSIN      IN      IP.TSP      TAPE STATUS IN
           RET
$TSOUT     OUT     OP.TSP      TAPE STATUS OUT
           RET
*
*
*      MISC PARAMS CONSTANTS
*
.CRCSUM    EQU      040027A
.CTC       EQU      002172A
.DLY       EQU      000053A
.RNB       EQU      002331A
.SRS       EQU      002265A
A.SYN      EQU      0260
A.STX      EQU      0020
.TPERR     EQU      002205A
.WNB       EQU      003024A
.WNP       EQU      003017A
ENDIF
IF      .UTILITY-55000A
*      USE DEBUG UTILITIES
$RCHAR     EQU      40144A
$WCHAR     EQU      40147A
$PRSCLE    RET                      SKIP CALL TO INITIALIZE PORTS -- HAN
DLED BY DEBUG
$CSINT     EQU      41043A
ENDIF
IF      A.CONSL
*      CONSOLE DRIVER NOT ASSEMBLED
ELSE
STL      'CONSOLE DRIVER'
EJECT
*
*      DATA AND BUFFERS
*
$INBUF     DB      0              INPUT BUFFER COUNT
           DS      30
$INBUFL    EQU      *-$INBUF-2    MAX LENGTH OF BUFFER
*
*      CONTROL CHARACTER TABLE
*
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

*          DB          CHAR, MASK, VALUE
*
$CSIB      DB          000, 3770, 0000    CTL-&
           DB          010, 1670, 2100    CTL-A
           DB          020, 1570, 2200    CTL-B
           DB          030, 1370, 2400    CTL-C
           DB          040, 0770, 3000    CTL-D
           DB          170, 1770, 0020    CTL-0
           DB          200, 1750, 0000    CTL-P
           DB          210, 1760, 0000    CTL-Q
           DB          230, 1760, 0010    CTL-S
           DB          1770                END OF TABLE
*
*          $CSIC--ADDRESS OF INTERRUPT TIME CONTROL CHARACTER PROCES-
SSOR
*
$CSIC      DW          $RET                ADDRESS OF USER ROUTINE FOR CTL-A TH
ROUGH      CTL-D
$CSLCTL    DB          0                  CONSOLE CONTROL BYTE
           SET         $INBUF/1000A
           ERRNZ       */1000A-.
           EJECT
*
*          $RCHAR  READ A SINGLE CHARACTER
*
*          ENTRY    NONE
*          EXIT      (A) = CHAR WITH PARITY BIT CLEANED
*          USES      A, F
*
$RCHAR     PUSH      H                    SAVE (HL)
           LXI        H, $INBUF           (HL)=ADDRESS OF CHARACTER POINTER
$RCHAR1    MOV        A, M                (A)=COUNTER
           ANA        A
           JZ         $RCHAR1             WAIT FOR INTERRUPT TO READ CHARACTER
           DI                    INTERLOCK SEQUENCE
           PUSH      D
           DCR        M                    DECREMENT COUNT
           MOV        D, M                (D)=COUNT -1
           INX        H
           MOV        A, M                (A)=READ CHARACTER
           CPI        1730                SEE IF LOWER CASE

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

JNC      $RC1.5
CPI      141Q
JC       $RC1.5      IS NOT LOWER CASE
SUI      40Q         MAKE UPPER CASE
$RCHARA  EQU      *-1 ZEROED FOR LOWER CASE
$RC1.5   PUSH     PSW  SAVE IT
$RCHAR2  DCR      D    MOVE OTHERS DOWN IN QUEUE
JM       $RCHAR3     NO MORE
INX      H
MOV      A,M
DCX      H
MOV      M,A
INX      H
JMP      $RCHAR2
$RCHAR3  EI          RESTORE INTERRUPTS
POP      PSW
POP      D
POP      H
RET
EJECT

*
*      $WCHAR -- WRITE SINGLE CHARACTER
*
*      ENTRY (A)=CHARACTER
*      EXIT  NONE
*      USES  NONE
$WCHAR   PUSH     PSW  SAVE CHARACTER
$WCHAR1  LDA      $CSLCTL CHECK CONTROL
ERRNZ    CC.HLD-1
RAR
JC       $WCHAR1     AM TO WAIT
RAR
ERRNZ    CC.DMP-2
JNC      $WCHAR2     AM TO PRINT
POP      PSW         CLT-0 IN EFFECT, DISCARD CHAR
RET

*      OVERFLOW TYPE-AHEAD BUFFER, ECHO BELL
$CSI2    DCR      M          REMOVE LAST CHARACTER INPUT TO MAKE
ROOM
MVI      A,BELL
PUSH     PSW

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

*      TYPE CHARACTER
$WCHAR2 CALL    $CISI
        ERRNZ    USR.TXR-1
        RAR
        JNC      $WCHAR2      WAIT FOR ROOM
        POP      PSW
        JMP      $CDOUT      OUTPUT AND EXIT
        EJECT
*      $PRSCl  PRESET CONSOLE DRIVERS
*
*      ENTRY  NONE
*      EXIT   NONE
*      USES   A,F
$PRSCl  MVI     A,2010
        CALL    $CISO      GUARANTEE OUT OF MODE-SET
        CALL    $TSOUT
        MVI     A,UCI.IR
        CALL    $CISO
        CALL    $TSOUT      FORCE INTO MODE-SET
        MVI     A,UMI.1B+UMI.LB+UMI.16%
        CALL    $CISO
        CALL    $TSOUT
        XRA     A
        STA     $INBUF
        MVI     A,3030      SET UP INTERRUPT VECTOR
        STA     .UIVEC+6
        LXI     H,$CSINT
        SHLD    .UIVEC+7
        MVI     A,UCI.ER+UCI.RE+UCI.IE+UCI.TE
        JMP     $CISO      ENABLE CONSOLE OUTPUT
        EJECT
*      $CSINT  CONSOLE INTERRUPT PROCESSOR
*
*      ENTRY  NONE
*      EXIT   EI AND RET
*      USES   NONE
$CSINT  PUSH    H
        PUSH    PSW
        CALL    $CISI
        ANI     USR.RXR
        CNZ     $CSI1      IF HAVE DATA FROM INPUT

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

$CSIX      POP      PSW
           POP      H
           EI
           RET
*          HAVE DATA INPUT INTERRUPT--SEE IF ROOM IN QUEUE
$CSI1      LXI      H,$INBUF
           MVI      A,$INBUFL
           CMP      M
           CC      $CSI2      NO ROOM
*          ADD CHARACTER TO QUEUE
$CSI3      INR      M
           MOV      A,M
           ADD      L
           MOV      L,A      (HL)=ADDRESS OF CHARACTER
           CALL     $CDIN     INPUT CHARACTER
           ANI      177Q     TRIM IT
           MOV      M,A
*          CHECK FOR SPECIAL CONTROL CHARACTERS
           CPI      '/'
           RNC      NOT CONTROL CHARACTER
*          HAVE CONTROL CHARACTER
           MVI      L,$CSIB-2
$CSI5      INX      H
           INX      H
           CMP      M      COMPARE TO CHARACTER
           RC      IS NOT IN LIST
           INX      H
           JNE      $CSI5     IS NOT THIS ONE
           LDA      $CSLCTL
           ANA      M      CLEAR $CSLCTL BITS
           INX      H
           XRA      M      SET $CTLCTL BITS
           STA      $CSLCTL
           PUSH     PSW
           MVI      L,$#INBUF
           DCR      M      DECREMENT CHARACTER FROM BUFFER
           POP      PSW
           RP      RETRUN IF NOT CTL-A THRU CTL-D
*          HAVE SPECIAL CONTROL CHAR--CALL USER
           LHLD     $CSIC
           PCHL      CALL USER ROUTINE

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

        ENDIF
        IF      A.CMPAR+A.LOPEN
.CMPAR   EQU      .ERMES
*        CMPAR NOT ASSEMBLED
        STL      'CMPAR'
        EJECT
        ELSE
        STL      'CMPAR'
        EJECT
*
*.CMPAR  COMPARE 2 CHARACTER STRINGS
*
*        <DE> = ADDRESS OF FIRST STRING
*        <HL> =ADDRESS OF SECOND STRING
*        <B> = NUMBER OF CHARACTERS TO COMPARE
.CMPAR   INR      B
        ANA      A          CLEAR CARRY FLAG
.CMPAR1  DCR      B
        RZ              STRING FINISHED -- MATCH
        LDAX   D          LOAD FIRST STRING
        CMP    M          COMPARE TO SECOND STRING ( D-H)
        RNZ              NO MATCH
        INX     D
        INX     H
        JMP     .CMPAR1
        ENDIF
        IF      A.FILL
.FILL    EQU      .ERMES
*        FILL NOT ASSEMBLED
        STL      'FILL'
        EJECT
        ELSE
        STL      'FILL'
        EJECT
*
*.FILL   MOVE FILL CHARACTERS INTO RECEIVING AREA
*
*        <A> = NUMBER OF FILL CHARACTERS TO MOVE TO RECEIVING ARE
*
*        <B> = FILL CHARACTER
*        <HL> = RECEIVING AREA
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```
*
.FILL      INR      A
.FILL1     DCR      A
           RZ
           MOV      M,B
           INX      H
           JMP      .FILL1
           ENDIF
           IF      A,LOKUP
.LOKLE     EQU      .ERMES
.LOKE      EQU      .ERMES
.LOKGE     EQU      .ERMES
.LOKL      EQU      .ERMES
*          TABLE LOOKUP NOT ASSEMBLED
           ELSE
           STL      'LOKUP'
           EJECT
*
*          LOOK UP ARGUMENT ON TABLE
*
*          ENTRY
*          (A)=NUMBER OF ARGUMENTS
*          (B)=LENGTH OF ARGUMENT
*          (C)=LENGTH OF ARGUMENT + RESULT
*          (DE)=ADDRESS OF SEARCH ARGUMENT
*          (HL)=ADDRESS OF TABLE
*
*          EXIT
*          (A)=INDEX OF MATCH ARG (>0)
*          NO MATCH (=0)
*          (B) UNCHANGED
*          (C) UNCHANGED
*          (DE) UNCHANGED
*          (HL)=ADDRESS OF RESULT (IF ANY)
*.LOKLE    LOOK UP LESS THAN OR EQUAL
.LOKLE     PUSH     PSW
           MVI      A,3120      JE
           STA      .LOKUPC
           MVI      A,3320      JC -- JUMP BORROW
           STA      .LOKUPC+3
           MVI      A,3770      SET P,NE FLAGS (0-(-1))
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

                STA      .LOKUP2+1
                JMP      .LOKUP
*.LOKE          LOOK UP EQUAL
.LOKE          PUSH     PSW
                MVI      A,3120          JE
                STA      .LOKUPC
                STA      .LOKUPC+3
                MVI      A,0010
                STA      .LOKUP2+1      SET NE FLAG
                JMP      .LOKUP
*.LOKGE         LOOK UP GREATER OR EQUAL
.LOKGE         PUSH     PSW
                MVI      A,3220          JNC -- JUMP NO BORROW
                STA      .LOKUPC
                STA      .LOKUPC+3
                MVI      A,0010          SET M FLAG
                STA      .LOKUP2+1
                JMP      .LOKUP
*.LOKL          LOOK UP LESS THAN
.LOKL          PUSH     PSW
                MVI      A,3320          JC -- JUMP BORROW
                STA      .LOKUPC
                STA      .LOKUPC+3
                MVI      A,3770          SET P FLAG
                STA      .LOKUP2+1
                JMP      .LOKUP
*
*.LOKUP         MAIN PROCESSING ROUTINE FOR TABLE LOOKUP SUBROUTINES
*
.LOKUP         POP      PSW
                INR      A
*
                COMPARISON LOOP
.LOKUP1        DCR      A
                JZ       .LOKUP2
                PUSH     B
                PUSH     D
                PUSH     H
                PUSH     PSW
                CALL     .CMPAR          SEARCH TABLE ARGUMENT
.LOKUPC        JE       .LOKUP3        FIRST TEST CONDITION
                JE       .LOKUP3        SECOND TEST CONDITION

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> 11

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```
*      NO MATCH YET
      POP      PSW
      MVI      B,0000      ZERO B OF (BC) PAIR
      POP      H
      DAD      B           INCREMENT TABLE ADDRESS
      POP      D
      POP      B
      JMP      .LOKUP1
*      NO MATCH -- ARGUMENTS EXHAUSTED
.LOKUP2 CPI      0010      SET RETURN CONDITION FLAG
      RET
*      MATCH -- POP REGS FROM STACK AND RETURN
.LOKUP3 MOV      C,B
      MVI      B,00
      DAD      B
      POP      D           POP PSW TO D
      MOV      A,D        MOVE INDEX TO A
      POP      D           POP H TO D
      POP      D           POP D TO D
      POP      B
      RET
      ENDIF
      IF      A.MOUBL
.LOUBLE EQU      .ERMES
*      MOUBL NOT ASSEMBLED
      STL      'MOUBL'
      EJECT
      STL      'MOUBL'
      EJECT
      ELSE
      STL      'MOVEBL'
      EJECT
*
*.MOUBL MOVE BLOCK OF CHARACTERS FROM SOURCE AREA TO DESTINATION
AREA
*
*      (DE)=ADDRESS OF SOURCE
*      (HL)=ADDRESS OF DESTINATION
*      (A)=NUMBER OF BYTES TO MOVE
*
*.MOUBL PUSH      B           SAVE CURRENT BC
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

      MOV     B,A           MOVE COUNT TO B REG
      INR     B
.MOUBL1  DCR     B           DECREASE COUNT
      JZ      .MOUBL2       RETURN IF BLCK MOVE COMPLETE
      LDAX    D             LOAD SOURCE CHARACTER
      MOV     M,A           MOVE TO DESTINATION AREA
      INX     D             INCREMENT D AND H REGS
      INX     H
      JMP     .MOUBL1
*        RETURN TO CALLING ROUTINE
.MOUBL2  POP     B
      RET
      ENDIF
      IF      A.SCAN
.MSCAN   EQU     .ERMES
*        SCAN NOT ASSEMBLED
      ELSE
      STL     'SCAN'
      EJECT
*.SCAN
*
*        ENTRY
*        (H,L)=ADDRESS OF STRING
*        (B)=MAX BYTES TO SEARCH
*        (A)=SEARCH CHARACTER
*
*        EXIT
*        (H,L)=ADDRESS OF MATCH OR NEXT STRING
*        (B)=0 NO MATCH
*        >0 MATCH -- NUMBER OF CHARS REMAINGING +1
*        C=MATCH BYTE IF MATCH
*        =MAX BYTES +1 IF NO MATCH
*
.MSCAN   INR     B
      MVI     C,0010
.MSCAN1  DCR     B
      RZ
      CMP     M
      RE
      INR     C
      INX     H
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

        JMP      .SCAN1
ENDIF
        IF       A.TYPE
.TYPE EQU      .ERMES
*        .TYPE NOT ASSEMBLED
ELSE
        STL      'TYPE'
EJECT
*
*.TYPE
*        PRINT CONTENTS OF BUFFER ON CONSOLE
*        (A)=NUMBER OF BYTES
*        (B,C)= LF BEFORE, AFTER LINE
*        (HL)=ADDRESS OF BUFFER
*
.TYPE   PUSH     D           SAVE D,E
        MOV      D,A
*        INITIAL LF/CR
        MVI      E,0000
        CALL     .TYPE3      INITIAL CR/LF
.TYPE1   DCR      D
        JM       .TYPE2
        MOV      A,M
        CALL     $WCHAR
        INX      H
        JMP      .TYPE1
.TYPE2   MOV      B,C
        MVI      E,0000
        CALL     .TYPE3
*        RETURN TO CALLING ROUTINE
        POP      D
        RET
*        CR/LF ROUTINE
.TYPE3   DCR      B
        JM       .TYPE31
        MVI      A,0120
        CALL     $WCHAR
        MOV      E,A
        JMP      .TYPE3
*        CARRIAGE RETURN
.TYPE31  MOV      A,E
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

        ANA      A
        JZ       .TYPE32
        MVI      A,0150
        CALL     $WCHAR
*       WAIT FOR RETURN -- 0.3 SEC
.TYPE32 MVI      A,300/2
        CALL     .DLY
        RET
        ENDIF
        IF      A.MOVDB
.MOVDB   EQU     .ERMES
        ELSE
        STL     'MOVDB'
        EJECT
*
*.MOVDB  MOVE LONG BLOCK ( GT 255 BYTES)
*
*       ENTRY
*       BC=NUMBER OF BYTES TO MOVE
*       DE=SOURCE AREA
*       HL=DESTINATION AREA
*
*
.MOVDB   PUSH    PSW
        MOV     A,B
        ANA     A
        JMP     .MOVDB2
.MOVDB1  MVI      A,3770
        CALL    .MOVBL
        MOV     A,C
        SUI     3770
        MOV     C,A
        MOV     A,B
.MOVDB2  SBI      0000
        MOV     B,A
        JNZ     .MOVDB1
        MOV     A,C
        CALL    .MOVBL
        POP     PSW
        RET
        ENDIF
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

      IF      A.DOPEN+A.DTAPE
.DOPEN EQU    .ERMES
*          DOPEN NOT ASSEMBLED
      ELSE
      STL     'DOPEN'
      EJECT

*.DOPEN
*
*          OPEN DUMP PORT FOR OUTPUT
*
*----- .DOPEN,.DWRITE,.DCLOSE -----
*          --ROUTINE FUNCTIONS
*              .DOPEN          WRITE LABEL RECORD
*                              INITIALIZE BLOCK COUNT
*
*              .DWRITE          COMPRESS BLANKS AND ADD EOL (000) (
0030 FILES ONLY)
*
*                              MOVE RECORD FROM WORK AREA TO OUTPUT
BLOCK
*
*                              UPDATE OUTPUT BLOCK COUNT
*                              WRITE BLOCK , IF FULL
*              .DCLOSE          WRITE PARTIAL BLOCK, IF NECESSARY
*                              WRITE EOF RECORD
*
*          --USER INITIALIZED UTILITY CONSTANTS -- LOCATED AT TOP
OF UTILITY PACKAGE
*              .DOPENF          FILE NAME
*              .DOPENR          ADDRESS OF OUTPUT BLOCK
*              .DOPENT          FILE TYPE 0030 (TEXT EDITOR COMPATIB
LE ASCII RECORDS) OR
*
*                              0110 (BYTE FORMAT -- ANY 8
BIT BYTE ACCEPTABLE)
*              .DOPENW          ADDRESS OF OUTPUT WORK AREA
*              .DOPENL          MAX LENGTH OF WOUTPUT BLOCK
*
*          --USER DEFINED AREAS -- LOCATED IN CALLING PROGRAM
*              OUTPUT BLOCK FORMAT:          2 BYTE BLOCK COUNT
*
*                              DATA BLOCK -- (.DOPEN
L) BYTES LONG
*
*              ADDRESS:          (.DOPENW)
*              CONTENTS          BLOCKED RECORDS TO BE
WRITTEN TO OUTPUT TAPE

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE

```

*
*          WORK AREA      FORMAT:          2 BYTE RECORD COUNT
*          DATA BLOCK -- LARGE E
* NOUGH TO HOLD LARGEST SINGLE RECORD
*          ADDRESS:          (.DOPENW)
*          CONTENTS:        RECORD TO BE MOVED TO
* OUTPUT BLOCK
*
*          --ERRORS  -- SOUND ALARM AND DISPLAY ERROR NUMBER
*          110        ATTEMPT TO OPEN FILE NOT CURRENTLY C
* LOSED , OR
*          TO CLOSE FILE NOT CURRENTLY
* OPEN
*
*          SAVE REGISTERS
* .DOPEN  PUSH   PSW
*          PUSH   B
*          PUSH   D
*          PUSH   H
*          CHECK STATUS
*          LDA     .DOPENS
*          CPI     'C'
*          MVI     A,1100
*          JNZ     .TPERR
*          CALL    .DOPEN0
*          BLOCKED
*          INITIALIZE RECORD COUNT
*          LHLD    .DOPENR
*          XRA     A
*          MOV     M,A
*          INX     H
*          MOV     M,A
*          RETURN
*          POP     H
*          POP     D
*          POP     B
*          POP     PSW
*          RET
*
*          WRITE BLOCK TO TAPE

```



PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

*
.DOPEN0 CALL .DHEAD
*      WRITE TYPE
*      LDA .DOPENT
*      CALL .WNB
*      WRITE SEQ=0
*      XRA A
*      STA .DOPENC
*      CALL .WNB
*      WRITE COUNT=HEADER RECORD LENGTH
*      LHD .DOPENF
*      CALL .WNP
*      WRITE LABEL
*      MOV B,L
*      LXI H,.DOPENF+2
.DOPEN1 MOV A,M
*      CALL .WNB
*      DCR B
*      INX H
*      JNZ .DOPEN1
*      WRITE CHECKSUM
*      LHD .CRCSUM
*      CALL .WNP
*      FLUSH CHECKSUM
*      CALL .WNP
*      STOP TRANSPORT
*      XRA A
*      OUT OP.TSP
*      UPDATE STATUS
*      MVI A,'0'
*      STA .DOPENS
*      RET
*      ENDIF
.DHEAD IF A.DTAPE+A.DOPEN+A.DWRIT+A.DCLOS
*      EQU .ERMES
*      ELSE
*      STL 'DHEAD'
*      EJECT
*.DHEAD
*      WRITE HEADER
*

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```
*      OUTPUT SYNC CHARS
.DHEAD  MVI    A,A.SYN
        MVI    B,32D
.DHEAD1  CALL   .WNB
        DCR    B
        JNZ    .DHEAD1
*      WRITE STX CHAR
        MVI    A,A.STX
        CALL   .WNB
*      CLEAR CRC SUM
        XRA    A
        MOV    H,A
        MOV    L,A
        SHLD   .CRC SUM
        RET
*
*      .DTAPE CONSTANTS
*
.DOPENS  DB     'C'           STATUS
.DOPENC  DB     0             SEQUENCE
        ENDIF
        IF     A.DWRIT+A.DTAPE
.DWRITE  EQU     .ERMES
.DWRIT0  EQU     .ERMES
*      DWRITE NOT ASSEMBLED
        ELSE
        STL     'DWRITE'
        EJECT
*.DWRITE
*
*      WRITE OUTPUT RECORD TO DUMP PORT
*
.DWRITE  PUSH    PSW
        PUSH    B
        PUSH    D
        PUSH    H
*      CHECK STATUS
        LDA     .DOPENS
        CPI     '0'
        MVI    A,1100
        JNZ    .TPERR
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```
.DWRBL   LDA     .DOPENL
        CPI     0030
        JE      .DWRCT0
        CPI     0110
        CNZ     .ERMES

*
*       PROCESS HEX FORMAT RECORD
*
*       MOVE WORK COUNT TO BC
.DWRBL0  LHL    .DOPENW
        MOV     C,M
        INX     H
        MOV     B,M
        LXI     H,2A
        DAD     B
*       SAVE WORK AREA COUNT
        SHLD    .DWRBL1+1
*       MOVE RECORD COUNT TO DE
        LHL    .DOPENR
        MOV     E,M
        INX     H
        MOV     D,M
*       CREATE POINTER AND SAVE
        INX     H                ADDRESS+2+CURRENT RECORD COUNT
        DAD     D
        SHLD    .DWRBL2+1
*       CREATE NEW RECORD COUNT AND SAVE
        LXI     H,2A
        DAD     B
        DAD     D                RECORD
        SHLD    .DWRBL3+1        NEW RECORD COUNT
*       COMPARE LENGTH WITH MAX
        LDA     .DOPENL
        SUB     L
        LDA     .DOPENL+1
        SBB     H
        JNC     .DWRBL1
*       WRITE CURRENT RECORD
        CALL    .DWRIT0
        LHL    .DOPENR
        XRA     A
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

        MOV     M,A
        INX     H
        MOV     M,A
        JMP     .DWRBL0
*      MOVE WORK AREA TO RECORD AREA
.DWRBL1 LXI     B,0A      WORK AREA COUNT
        LHLD    .DOPENW   WORK AREA ADDRESS TO DE
        XCHG
.DWRBL2 LXI     H,0A      POINTER
        CALL    .MOUDB
.DWRBL3 LXI     D,0A      NEW RECORD COUNT
        LHLD    .DOPENR
        MOV     M,E
        INX     H
        MOV     M,D
*      RETURN
.DWRBL4 POP     H
        POP     D
        POP     B
        POP     PSW
        RET
*
*      PROCESS COMPRESSED TEXT (CT) RECORD
*
*      COMPRESS OUT BLANKS AND RECORD COUNT FROM WORK AREA -- A
DD EOL (0000)
.DWRCT0 LHLD    .DOPENW
        MOV     C,M
        INX     H
        MOV     B,M
        INX     H
        XCHG     WORK AREA COUNT TO (B,C)
        XCHG     WORK AREA ADDRESS TO D -- FIRST DATA
BYTE
        LHLD    .DOPENW   WORK AREA ADDRESS TO H -- FIRST BYTE
OF COUNTER
*      STORE DEFAULT BLANK
.DWRCT1 MVI     M,2000
*      LOAD NEXT BYTE
        DCX     B
        MOV     A,B
        ANA     A

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: UTILITIES - PRICE < CONT'D >

```

        JM      .DWRCT4+1      MAX LENGTH RECORD 32K
        LDAX    D
        CPI     0400
        JE      .DWRCT3
*      MOVE CHARACTER
.DWRCT2 MOV      M,A
        INX     D
        INX     H
        JMP     .DWRCT1      SET UP DUMMY BLANK AGAIN
*      MOVE BLANK(S)
.DWRCT3 INR      M
        INX     D
        DCX     B
        MOV     A,B
        ANA     A
        JM      .DWRCT4
        LDAX    D
        CPI     0400
        JE      .DWRCT3
        INX     H
        JMP     .DWRCT2
.DWRCT4 INX     H
        MVI     M,0000
*
*      SET UP NEW RECORD COUNT, POINTER, ETC
*
*      WORK AREA COUNT IN (B,C)
*      XCHG      ADDRESS OF LAST BYTE TO (D,E)
        LHLD    .DOPENW
        MOV     A,E
        SUB     L
        MOV     C,A
        MOV     A,D
        SBB     H
        MOV     B,A
        INX     B
*      MOVE WORK AREA COUNT (B,C) TO (H,L)
        MOV     H,B
        MOV     L,C
        SHLD    .DWRBL1+1
*      MOVE RECORD COUNT TO (D,E)
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER

\* SL-TAPEM

TITLE 'TAPE MANAGEMENT

VERSION 00.00 780503'

STL 'MAIN ROUTINE

BY: RON TURNER'

EJECT

ORG 40100A

START JMP COLD

JMP WARM

COLD CALL TPDATA

SPACE 2

\* PROMPT OPERATOR

WARM LXI D,PROMPT

MVI B,9

CALL WC.RTN

\* READ COMMAND

LXI H,INPUT

MVI B,20

CALL RC.RTN

\* SEARCH COMMAND TABLE

LXI D,INPUT

LXI H,TABLE

MOV B,C

CALL FC.RTN

\* DONE, GO PROMPT OPERATOR

JMP WARM

EJECT

\* SUBROUTINE - READ A TAPE RECORD

READ LXI H,TPAREA

LXI B,TPLNTH

CALL RT.RTN

LXI D,TPFMT

LXI H,DMFMT

LDAX D

ANI 1770

MOV M,A

INX D

INX H

LDAX D

INR A

MOV M,A

CALL DSPLY1

CALL DSPLY2

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        LDA    TPFMT
        RLC
        JC     PRTEOF1
        LXI    D,FEED
        MUI    B,2
        CALL   WC.RTN
        RET
PRTEOF1 LXI    D,EOFMSG
        MUI    B,6
        CALL   WC.RTN
        XRA    A
        STA    DMSEQ
        RET

SPACE 2
* SUBROUTINE - WRITE A TAPE RECORD
WRITE   LXI    D,TPAREA
WRTCALL PUSH   D
        CALL   WT.RTN
* ESTABLISH NEXT SEQUENCE NUMBER
        POP    D
        LDAX   D
        RLC
        JC     WRTEOF
        INX    D
        LDAX   D
        INR    A
        JMP     WRTMOV
WRTEOF  XRA    A
WRTMOV  STA    DMSEQ
* NOTIFY OPERATOR - DONE
WRTDONE LXI    D,DONE
        MUI    B,11
        CALL   WC.RTN
        RET

EJECT
* SUBROUTINE - COPY TAPE RECORDS
* (BC) = PARAMETER LENGTH
COPY    MOV    C,B
        XRA    A
        MOV    B,A
* PARAMETER DEFAULT 1
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        ORA      C
        JNZ      CUERT
        LXI      H,1
        SHLD     RECCNT
        JMP      TSTCNT
* (DE) -> END (LSB) OF PARAMETER
CUERT   XCHG
        DAD      B
        XCHG
        DCX      D
* -RECCNT- = BINARY PARAMETER
        MOV      B,C
        LXI      H,RECCNT
        MVI      C,2
        CALL     CB.RTN
* -RECCNT- = 0?
TSTCNT  LHLD     RECCNT
        MOV      A,H
        ORA      L
        RZ
        DCX      H
        SHLD     RECCNT
        LXI      H,FIRSTTP
        SHLD     CPYSWT+1
* READ A TAPE RECORD
CPREAD  LXI      H,TPAREA
        LXI      B,TPLNTH
        CALL     RT.RTN
        LXI      D,TFMT
        LXI      H,DMFMT
        LDAX     D
        ANI      1770
        MOV      B,A
        STA      BINFMTH
        INX      H
        MOV      A,M
        DCX      H
        ANA      A
        JNZ      NOTLEBL
        MOV      M,B          INIT DMFMT
NOTLEBL MOV      A,B
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

                                CMP     M
                                INX     D
                                INX     H
                                LDAX    D
                                STA     BINSEQN
                                JNZ     CPYERR
                                CMP     M
                                JNZ     CPYERR

EJECT
CPYSWT      JMP     @
FIRSTTTP   LXI     H, NXTSEQ
           SHLD    CPYSWT+1
           CALL   DSPLY1
NXTSEQ     CALL   DSPLY2
           LXI     D, TPAREA
           CALL   WT, RTN
           LXI     H, DMSEQ
           LDA     TPFMT
           RLC
           JC      PRTEOF
           INR     M
           LXI     D, SEMICLN
           MUI     B, 1
           CALL   WC, RTN
           JMP     CPREAD
PRTEOF     MUI     M, @          ZERO DMSEQ
           LXI     D, EOFMSG
           MUI     B, 6
           CALL   WC, RTN
           JMP     TSTCNT
CPYERR     LXI     D, BINFMTH
           LXI     H, ASFMTH2
           MUI     B, 1
           MUI     C, 3
           PUSH    B
           CALL   BC, RTN
           LXI     D, BINSEQN
           LXI     H, ASSEQN2
           POP     B
           CALL   BC, RTN
           LXI     D, CPYMSG1

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

      MVI      B,CPYSIZ1
      CALL     WC.RTN
      CALL     AO.RTN
      RET
EJECT
DSPLY1  LXI      D,DMFMT
        LXI      H,FMTMSG2
        MVI      B,1
        MVI      C,3
        CALL     BC.RTN
        LXI      D,FMTMSG1
        MVI      B,11
        CALL     WC.RTN
        LDA      TPSEQ
        ANA      A
        JNZ      PRTREC
        LXI      D,LEBLEQ
        MVI      B,6
        CALL     WC.RTN
        LXI      D,TPDATA
        LXI      H,CNSLBL
        MVI      B,63
        CALL     MA.RTN
        LXI      H,TPSIZE
        MOV      D,M
        INX      H
        MOV      E,M
        LXI      B,63
        MOV      A,B
        CMP      D
        JC       FULLBL
        MOV      A,C
        CMP      E
        JC       FULLBL
FULLBL  MOV      C,E
        MOV      A,C
        ANA      A
        JZ       PRTFEED
        LXI      D,CNSLBL
        MOV      B,C
        CALL     WC.RTN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
PRTFEED  LXI      D,FEED
          MUI      B,2
          CALL     WC.RTN
PRTREC   LXI      D,RECMMSG
          MUI      B,5
          CALL     WC.RTN
          RET

SPACE 2
DSPLY2   LXI      D,TPSEQ
          LXI      H,CNSSEQ2
          MUI      B,1
          MUI      C,3
          CALL     BC.RTN
          LXI      D,CNSSEQ1
          MOV      B,C
          CALL     WC.RTN
          RET

EJECT
* SUBROUTINE - DUMP TAPE
LIST0    LXI      H,MOVEQ
          MUI      A,12
          JMP      LIST1
LISTX    LXI      H,MOVEX
          MUI      A,16
LIST1    SHLD     LSWT+1
          STA     LNSIZE+1
          LXI      D,LNFEED
          MUI      B,1
          CALL     WC.RTN
          LXI      H,1
          SHLD     TPCNT
          LXI      H,TPAREA
          LXI      B,TPSIZE
          LDAX     B
          MOV      D,A
          INX      B
          LDAX     B
          MOV      E,A
          INX      D
          INX      D
          INX      D
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

	INX	D
	LDA	TPFMT
	ANI	1770
	CPI	1
	JNZ	LIST2
	INX	D
	INX	D
	INX	D
	INX	D
LIST2	PUSH	D
	PUSH	H
	LXI	H,CNSCHR
	MVI	B,69
	CALL	BM.RTN
LNSIZE	LXI	B,0
	MOV	A,B
	CMP	D
	JC	FULINE
	MOV	A,C
	CMP	E
	JC	FULINE
	MOV	C,E
FULINE	PUSH	B
	LXI	D,TPCNT
	LXI	H,CNSLSB1
	MVI	B,2
	MVI	C,4
	CALL	BC.RTN
	POP	B
	PUSH	B
	LHLD	TPCNT
	DAD	B
	SHLD	TPCNT
EJECT		
	DCX	H
	SHLD	BINARY2
	LXI	D,BINARY2
	LXI	H,CNSLSB2
	MVI	B,2
	MVI	C,4
	CALL	BC.RTN

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
LSWT      POP      B
          POP      D
          PUSH     D
          LXI      H,CNSCHR
          MOV      B,C
          CALL     MA.RTN
          MVI      M,'/'
          POP      H
          POP      D
          PUSH     H
          PUSH     D
          PUSH     B
          CALL     @
          POP      B
          POP      D
          MVI      M,150
          INX      H
          MVI      M,120
          INX      H
          PUSH     D
          LXI      D,CNSOUT
          MOV      A,L
          SUB      E
          MOV      B,A
          CALL     WC.RTN
          POP      D
          MOV      A,E
          SUB      C
          MOV      E,A
          MOV      A,D
          SBB      B
          MOV      D,A
          ORA      E
          POP      H
          RZ
          MOV      A,L
          ADD      C
          MOV      L,A
          MOV      A,H
          ADC      B
          MOV      H,A
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

      JMP      LIST2
EJECT
MOVEX      XCHG
           LXI      H,CNSEXP
MOVEX1     MVI      B,8
MOVEX2     CALL     XH.RTN
           DCR      C
           RZ
           INX      H
           DCR      B
           JNZ      MOVEX2
           INX      H
           INX      H
           JMP      MOVEX1
MOVEQ      XCHG
           LXI      H,CNSEXP
MOVEQ1     MVI      B,6
MOVEQ2     CALL     XO.RTN
           DCR      C
           RZ
           INX      H
           DCR      B
           JNZ      MOVEQ2
           INX      H
           INX      H
           JMP      MOVEQ1
* SUBROUTINE - WRITE LABEL RECORD
LABEL      XRA      A
           STA      DMSEQ
           MOV      A,B
           STA      DMSIZ+1
           ANA      A
           JZ       WRTCALL
           LXI      H,DMDATA
           CALL     MV.RTN
           LXI      D,DMREC
           JMP      WRTCALL
* SUBROUTINE - WRITE DUMMY SEQ# RECORD
DUMMY      MOV      C,B
           XRA      A
           MOV      B,A
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        XCHG
        DAD    B
        XCHG
        DCX    D
        MOV    B,C
        LXI    H,DMSEQ
        MVI    C,1
        CALL   CB.RTN
        MVI    A,0
        STA    DMSIZ+1
        LXI    D,DMREC
        JMP    WRTCALL

EJECT
* SUBROUTINE - CHANGE SEQ#
SEQ      MOV    C,B
        XRA    A
        MOV    B,A
        XCHG
        DAD    B
        XCHG
        DCX    D
        MOV    B,C
        LXI    H,TPSEQ
        MVI    C,1
        CALL   CB.RTN
        JMP    WRTDONE

EJECT
* SUBROUTINE - WRITE EOF RECORD
* PARAMETER LENGTH 0?
EOF      MOV    A,B
        ANA    A
        RZ
* FIND FORMAT PARAMETER
        MVI    C,0
FNDFMT   INX    D
        INR    C
        DCR    B
        RZ
        LDAX   D
        CPI    / /
        JNZ    FNDFMT
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
* MOVE FORMAT TO OUTPUT
      PUSH    B
      PUSH    D
      DCX     D
      LXI     H,EOFREC
      MOV     B,C
      MVI     C,1
      CALL    CB.RTN
* SET EOF BIT IN OUTPUT
      MOV     A,M
      ORI     2000
      MOV     M,A
* NO SEQ# PARAMETER?
      POP     D
      POP     B
      DCR     B
      RZ
* FIND SEQ# PARAMETER
      MVI     C,0
FNDSEQ INX     D
      INR     C
      DCR     B
      JNZ     FNDSEQ
* MOVE SEQ# TO OUTPUT
      INX     H
      MOV     B,C
      MVI     C,1
      CALL    CB.RTN
* WRITE EOF RECORD
      LXI     D,EOFREC
      JMP     WRTCALL
* SUBROUTINE - COMMAND LIST
HELP   LXI     D,HLPMSG1
      MVI     B,HLPsiz1
      CALL    WC.RTN
      LXI     D,HLPMSG2
      MVI     B,HLPsiz2
      CALL    WC.RTN
      RET
* EL-TAPEM
* SL-AD.
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
STL 'ASCII TO DECIMAL          VERSION 00.00 780417'
EJECT
AD.DCWK DS      4              BY: RON TURNER
SPACE 2
* SAVE TARGET LENGTH, TARGET ADDRESS
AD.RTN  PUSH    B
        PUSH    H
* (C) = SOURCE LENGTH; ZERO WORK AREA
        MOV     C,B
        LXI     H,AD.DCWK
        MVI     B,4
        CALL    ZM.RTN
* INITIALIZE FLIP FLOP
        MVI     B,1
* MOVE SOURCE TO WORK
        LXI     H,AD.DCWK-1
AD.NXTA LDAX    D
        ANI     170
        DCR     B
        JZ      AD.FXA1
        RLC
        RLC
        RLC
        RLC
        JMP     AD.FXA2
AD.FXA1 INX     H
        MVI     B,2
AD.FXA2 ORA     M
        MOV     M,A
        DCX     D
        DCR     C
        JNZ     AD.NXTA
* MOVE WORK TO TARGET
        LXI     D,AD.DCWK
        POP     H
        POP     B
        MOV     B,C
        CALL    MV.RTN
        RET
* EL-AD.
* SL-AD.
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
STL 'ALERT OPERATOR          VERSION 00.00 780310'
EJECT
AO.BELS  DB      70,70,70,70,70,70  BY: RON TURNER
AO.TICK  EQU     40033A
SPACE 2
* NEGATE CTL-A & CTL-O
AO.RTN   LDA     CI.CTL
        ANI     1650
        STA     CI.CTL
* CYCLE CONSOLE BELL
AO.RING  LXI     D,AO.BELS
        MVI     B,6
        LDA     AO.TICK+1
        RRC
        CC      WC.RTN
* TEST FOR CTL-A
        LDA     CI.CTL
        RRC
        RRC
        RRC
        RRC
        JNC     AO.RING
* GOT CTL-A, CLEAR CTL-A & RETURN
        LDA     CI.CTL
        ANI     1670
        STA     CI.CTL
        RET
```

\* EL-AO.

\* SL-BA.

```
STL 'BINARY ADDITION        VERSION 00.00 780312'
EJECT
BA.RTN   ANA     A                      BY: RON TURNER
BA.ADD   LDAX   D
        ADC     M
        MOV     M,A
        INX     D
        INX     H
        DCR     B
        JNZ     BA.ADD
        RET
```

\* EL-BA.

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
* SL-BC.
  STL 'BINARY TO ASCII          VERSION 00.00 780413'
  EJECT
BC.DEC  DS      4                      BY: RON TURNER
  SPACE 2
BC.RTN  PUSH    B
        PUSH    H
        LXI     H,BC.DEC
        PUSH    H
        MVI     C,4
        CALL    BD.RTN
        POP     D
        POP     H
        POP     B
        MOV     B,C
        CALL    DA.RTN
        RET

* EL-BC.
* SL-BD.
  STL 'BINARY TO DECIMAL        VERSION 00.00 780403'
  EJECT
* TABLE OF BINARY POWERS OF TEN      BY: RON TURNER
*      DB      LSB.....MSB
BD.TBL  DB      2000,2260,2300 TEN MILLION
        DB      1000,1020,0170 ONE MILLION
        DB      2400,2060,0010 ONE HUNDRED THOUSAND
        DB      0200,0470,0000 TEN THOUSAND
        DB      3500,0030,0000 ONE THOUSAND
        DB      1440,0000,0000 ONE HUNDRED
        DB      0120,0000,0000 TEN
        DB      0010,0000,0000 ONE
        DB      0000                      END OF TABLE

* WORK AREAS
BD.BNWK DS      3
BD.DCWK DS      4
BD.DCHI EQU     *-1
        ERNZ    *-BD.BNWK-7

  SPACE 2
* SAVE TARGET ADDRESS
BD.RTN  PUSH    H
* SAVE SOURCE LENGTH, TARGET LENGTH
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
        PUSH    B
* ZERO WORK AREAS
        LXI     H,BD.BNWK
        MVI     B,7
        CALL    ZM.RTN
* RETRIEVE SOURCE LENGTH
        POP     B
* RESTORE TARGET LENGTH
        PUSH    B
* MOVE SOURCE TO BINARY WORK AREA
        LXI     H,BD.BNWK
        CALL    MV.RTN
* INITIALIZE AND SAVE DECIMAL WORK POINTER
        LXI     H,BD.DCHI+1
        PUSH    H
* INITIALIZE AND SAVE FLIP FLOP
        MVI     B,2
        PUSH    B
* INITIALIZE TABLE INDEX
        LXI     H,BD.TBL
* INITIALIZE COUNTER
BD.SUB1  MVI     C,0
* SUBTRACT A POWER OF TEN FROM BINARY WORK
BD.SUB2  LXI     D,BD.BNWK
        MVI     B,3
        ANA     A
BD.SUB3  LDAX    D
        SBB     M
        STAX    D
        DCR     B
        JZ      BD.CNT
        INX     H
        INX     D
        JMP     BD.SUB3
EJECT
* (C) = # OF TIMES POWER SUBTRACTED
BD.CNT   INR     C
        DCX     H
        DCX     H
        JNC     BD.SUB2
* HAVE BORROW, UNDO LAST SUBTRACT
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
      DCR      C
* (DE) -> POWER
      XCHG
      LXI      H,BD.BNWK
      MVI      B,3
      CALL     BA.RTN
* PREPARE (C) FOR MOVE TO DECIMAL WORK AREA
      MOV      A,C
      POP      B
      POP      H
      DCR      B
      JZ       BD.DCS1
      RLC
      RLC
      RLC
      RLC
      DCX      H
      JMP      BD.DCS2
BD.DCS1 MVI      B,2
BD.DCS2 PUSH    H
      PUSH    B
* MOVE (C) TO DECIMAL WORK AREA
      ORA      M
      MOV      M,A
* (HL) -> NEXT POWER OF TEN
      XCHG
* TEST FOR END OF TABLE
      MOV      A,M
      ANA      A
      JNZ      BD.SUB1
* FLUSH STACK; MOVE DECIMAL WORK TO TARGET
      POP      H
      POP      H
* (B) = TARGET LENGTH
      POP      B
      MOV      B,C
* (HL) -> TARGET
      POP      H
* FILL TARGET
      LXI      D,BD.DCWK
      CALL     MV.RTN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

      RET
* EL-BD.
* SL-BM.
  STL 'BLANK MEMORY'          VERSION 00.00 780227'
  EJECT
BM.RTN   MUI      M, 4
        DCR      B
        RZ
        INX      H
        JMP      BM.RTN
* EL-BM.
* SL-CB.
  STL 'ASCII TO BINARY'      VERSION 00.00 780419'
  EJECT
CB.DCML  DS       4          BY: RON TURNER
CB.MSB   EQU      *-1
  SPACE 2
CB.RTN   PUSH     B
        PUSH     H
        LXI      H, CB.DCML
        MUI      C, 4
        CALL     AD.RTN
        LXI      D, CB.MSB
        POP      H
        POP      B
        MUI      B, 4
        CALL     DB.RTN
        RET
* EL-CB.
* SL-CI.
  STL 'CONSOLE INTERRUPTS'   VERSION 00.01 780501'
  EJECT
* CONSOLE INPUT QUEUE          BY: RON TURNER
CI.QSZ   DB       0
        DS       30
CI.QMAX  EQU      *-CI.QSZ-2
* CONTROL CHARACTER TABLE
*          DB      CHAR, MASK, VALUE
* IF CHAR:
*   -CI.CTL- = (-CI.CTL- AND -MASK-) XOR -VALUE-
CI.CTTB  DB       000, 3770, 0000  CTL=@ (NULL)
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        DB      010,1670,2100  CTL-A
        DB      020,1570,2200  CTL-B
        DB      030,1370,2400  CTL-C
        DB      040,0770,3000  CTL-D
        DB      170,1770,0020  CTL-O
        DB      200,1750,0000  CTL-P
        DB      210,1760,0000  CTL-Q
        DB      230,1760,0010  CTL-S
        DB      1770          END OF TABLE
* USER INTERRUPT HANDLER ADDRESS
CI.UINT  DW      CI.RET
* CONSOLE CONTROL BIT STORAGE
CI.CTL   DB      0
CI.BELL  EQU     70          OUTPUT TO RING BELL
CI.CSDT  EQU     3720       CONSOLE DATA PORT
CI.CSST  EQU     3730       CONSOLE STATUS PORT
        SPACE 2
CI.RTN   PUSH    H
        PUSH    PSW
* TEST CONSOLE STATUS FOR DATA AVAILABLE
        IN      CI.CSST
        ANI     20
* HAVE DATA, CALL ROUTINE TO ADD TO QUEUE
        CNZ     CI.READ
CI.RET   POP      PSW
        POP     H
        EI
        RET
* TEST FOR ROOM IN QUEUE FOR DATA
CI.READ  LXI     H,CI.QSZ
        MVI     A,CI.QMAX
        CMP     M
* QUEUE FILLED, CALL OVERFLOW ROUTINE
        CC      CI.OFLO
        EJECT
* ADD BYTE TO QUEUE
        INR     M
        MOV     A,L
        ADD     M
        MOV     L,A
        MOV     A,H
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        ACI      0
        MOV      H,A
        IN       CI.CSDT
        ANI      1770
        MOV      M,A
* TEST FOR CONTROL CHARACTER
        CPI      / /
        RNC
* HAVE CONTROL CHARACTER, SEARCH TABLE
        LXI      H,CI.CTTB-2
CI.LOOK  INX      H
        INX      H
        CMP      M
        RC              END OF TABLE
        INX      H
        JNE      CI.LOOK
* FOUND TABLE ENTRY, CLEAR CONSOLE CONTROL BITS
        LDA      CI.CTL
        ANA      M
        INX      H
* SET CONSOLE CONTROL BITS AND FLAG
        XRA      M
        STA      CI.CTL
* SAVE CONSOLE CONTROL BITS AND FLAG
        PUSH     PSW
* REMOVE CHARACTER FROM QUEUE
        LXI      H,CI.QSZ
        DCR      M
* RETURN IF NOT CTL-A THRU CTL-D
        POP      PSW
        RP
* HAVE USER INTERRUPT, GO TO USER ROUTINE
        LALD     CI.UINT
        PCHL
* QUEUE OVERFLOW, DISCARD LAST BYTE, RING BELL
CI.OFLO  DCR      M
CI.WAIT  IN       CI.CSST
        RAR
        JNC      CI.WAIT
        MVI      A,CI.BELL
        OUT      CI.CSDT
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

                RET
* EL-CI.
* SL-DA.
  STL 'DECIMAL TO ASCII'          VERSION 00.00 780406'
  EJECT
DA.RTN  LDAX    D                      BY: RON TURNER
        ANI     170
        CALL    DA.CURT
        RZ
        LDAX    D
        ANI     3600
        RRC
        RRC
        RRC
        CALL    DA.CURT
        RZ
        INX     D
        JMP     DA.RTN
DA.CURT ADI     600
        MOV     M,A
        DCX     H
        DCR     B
        RET
* EL-DA.
* SL-DB.
  STL 'DECIMAL TO BINARY'         VERSION 00.00 780419'
  EJECT
DB.FLIP DS      1                      BY: RON TURNER
DB.HLD1 DS      3
DB.HLD2 DS      3
  SPACE 2
* SAVE TARGET ADDRESS; SOURCE & TARGET LENGTH
DB.RTN  PUSH    H
        PUSH    B
* ZERO TARGET & HOLD AREAS
        MOV     B,C
        CALL    ZM.RTN
        LXI     H,DB.HLD1
        MVI     B,6
        CALL    ZM.RTN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
* INITIALIZE FLIP FLOP
      MVI      A,01010101B
      STA      DB.FLIP
* (B) = SOURCE LENGTH + 1
      POP      B
      POP      H
      INR      B
* FIND BCD DIGIT
DB.NEXT LDA      DB.FLIP
      RLC
      STA      DB.FLIP
      JC       DB.FXA1
* TEST FOR END OF SOURCE
      DCR      B
      RZ
* HI ORDER 4 BITS
      LDAX     D
      ANI      3600
      RRC
      RRC
      RRC
      RRC
      JMP      DB.FXA2
* LO ORDER 4 BITS
DB.FXA1 LDAX     D
      DCX      D
      ANI      170
      EJECT
* PREPARE TO ADD DIGIT TO OUTPUT
DB.FXA2 STA      DB.HLD1
      PUSH     D
      PUSH     B
      MOV      B,C
      PUSH     B
      PUSH     H
* MOVE TARGET TO HOLD
      XCHG
      LXI      H,DB.HLD2
      CALL     MV.RTN
* TARGET = TARGET * 4
      POP      H
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
      POP      B
      PUSH     B
      PUSH     H
      CALL     SL.RTN
      POP      H
      POP      B
      PUSH     B
      PUSH     H
      CALL     SL.RTN
* TARGET = HOLD * 5
      LXI      D,DB.HLD2
      POP      H
      POP      B
      PUSH     B
      PUSH     H
      CALL     BA.RTN
* TARGET = TARGET * 2
      POP      H
      POP      B
      PUSH     B
      PUSH     H
      CALL     SL.RTN
* TARGET = TARGET + DIGIT
      LXI      D,DB.HLD1
      POP      H
      POP      B
      PUSH     H
      CALL     BA.RTN
* GET NEXT DIGIT
      POP      H
      POP      B
      POP      D
      JMP      DB.NEXT
* EL-DB.
* SL-FC.
      STL 'FIND COMMAND'          VERSION 00.00 780331'
      EJECT
FC.BMSG DB 'INVALID. REENTER ' BY: RON TURNER
      SPACE 2
* BYPASS INSIGNIFICANT COMMAND BLANKS
FC.RTN  CALL FC.BYFB
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
* TEST FOR ZERO COMMAND LENGTH
      JZ      FC.BAD
* SAVE COMMAND LENGTH, COMMAND ADDRESS
FC.SBDE  PUSH  B
        PUSH  D
* TEST FOR SIGNIFICANT COMMAND BLANK
FC.TSTB  LDAX  D
        CPI   / /
        JZ    FC.TSTT
* SEE IF MORE TABLE ENTRY TO COMPARE
        MOV   A,M
        ANA   A
        JZ    FC.BYPT
* COMPARE A COMMAND BYTE TO A TABLE BYTE
        LDAX  D
        CMP   M
        JNZ   FC.BYPT
* BYTES EQUAL, PREPARE NEXT COMPARE
        INX   D
        INX   H
        DCR   B
* TEST FOR END OF COMMAND
        JNZ   FC.TSTB
* SEE IF TABLE ENTRY ALL COMPARED
FC.TSTT  MOV   A,M
        ANA   A
        JNZ   FC.BYPT
* BYPASS COMMAND BLANKS TO PARAMETERS
        CALL  FC.BYPB
* STORE PARAMETER ADDRESS
        PUSH  D
EJECT
* (HL) = ROUTINE ADDRESS
        INX   H
        MOV   E,M
        INX   H
        MOV   D,M
        XCHG
* RESTORE PARAMETER ADDRESS
        POP   D
* RESTORE STACK
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        INX     SP
        INX     SP
        INX     SP
        INX     SP
* GO TO ROUTINE
        PCHL
* NOT THIS TABLE ENTRY, TRY NEXT ONE
FC.BYPT  POP     D
        POP     B
FC.TST0  MOV     A,M
        ANA     A
        JZ      FC.NXTT
        INX     H
        JMP     FC.TST0
FC.NXTT  INX     H
        INX     H
        INX     H
* TEST FOR END OF TABLE
        MOV     A,M
        ANA     A
        JZ      FC.BAD
        JMP     FC.SBDE
* BYPASS BLANKS IN COMMAND
FC.BYPB  MOV     A,B
        ANA     A
        RZ
        LDAX   D
        CPI    / /
        RNZ
        INX     D
        DCR     B
        JMP     FC.BYPB
* COMMAND NOT FOUND, NOTIFY OPERATOR
FC.BAD   LXI     D,FC.BMSG
        MVI     B,17
        CALL    WC.RTN
        RET
* EL-FC.
* SL-NA.
        STL    MOVE ASCII CHARACTERS  VERSION 00.00 780303'
        EJECT
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

MA.RTN      LDAX      D                      BY: RON TURNER  
             CPI      400  
             JC      MA.BLNK  
             CPI      1770  
             JC      MA.MOVE  
MA.BLNK      MVI      A,      '  
MA.MOVE      MOV      M,A  
             INX      D  
             INX      H  
             DCR      B  
             JNZ      MA.RTN  
             RET

\* EL-MA.

\* SL-MV.

STL 'MOVE STRING  
EJECT

VERSION 00.00 780302'

MV.RTN      LDAX      D                      BY: RON TURNER  
             MOV      M,A  
             INX      D  
             INX      H  
             DCR      B  
             JNZ      MV.RTN  
             RET

\* EL-MV.

\* SL-PI.

STL 'PROCESS INTERRUPTS  
EJECT

VERSION 00.00 780512'

\* TEST FOR CTL-C

BY: RON TURNER

PI.RTN      LDA      CI.CTL  
             RLC  
             RLC  
             RLC  
             RNC

\* GOT CTL-C, WARM START

LHLD      PS.STAK  
SPHL  
XRA      A  
STA      CI.CTL  
EI  
JMP      COLD

\* EL-PI.

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
* SL-PS.
  STL 'PRESET I/O OPERATIONS  VERSION 00.00 780224'
  EJECT
*
                                     BY: RON TURNER
PS.TPST  EQU      3710              TAPE STATUS
PS.CSST  EQU      3730              CONSOLE STATUS
PS.I03   EQU      40045A           PAM-8 UIVEC+9
PS.STAK  DW       0                STACK POINTER STORAGE
  SPACE 2
* STORE STACK POINTER
PS.RTN   LHLD     PS.STAK
        MOV      A,H
        ORA      L
        JNZ      PS.MODE
        DAD      SP
        SHLD     PS.STAK
* GUARANTEE OUT OF MODE-SET
PS.MODE  MVI      A,2010
        OUT      PS.CSST
        OUT      PS.TPST
* FORCE INTO MODE-SET
        MVI      A,1000
        OUT      PS.CSST
        OUT      PS.TPST
        MVI      A,1160
        OUT      PS.CSST
        OUT      PS.TPST
* ENABLE CONSOLE OUTPUT
        MVI      A,270
        OUT      PS.CSST
* SET UP CONSOLE INTERRUPT ENTRY POINT
        MVI      A,3030
        STA      PS.I03
        LXI      H,CI.RTN
        SHLD     PS.I03+1
        RET
* EL-PS.
* SL-RC.
  STL 'READ CONSOLE  VERSION 00.00 780304'
  EJECT
*
                                     BY: RON TURNER
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
RC.CSDT EQU 3720      CONSOLE DATA
RC.CSST EQU 3730      CONSOLE STATUS
SPACE 2
* ZERO OUTPUT (C)
RC.RTN XRA A
      MOV C,A
* TEST QUEUE SIZE FOR PRESENCE OF DATA
RC.CKQ PUSH H
      LXI H,CI.QSZ
RC.NYET MOV A,M
      ANA A
      JZ RC.NYET
* DATA IN QUEUE, PROCESS FIRST BYTE FROM QUEUE
      DI
* DECREMENT QUEUE SIZE, (D) = QUEUE SIZE FOR SHIFT
      DCR M
      MOV D,M
* (A) = FIRST BYTE FROM QUEUE
      INX H
      MOV A,M
* SHIFT REMAINING QUEUE BYTES LEFT ONE
RC.SHFT DCR D
      JM RC.NABL
      INX H
      MOV E,M
      DCX H
      MOV M,E
      INX H
      JMP RC.SHFT
RC.NABL EI
      POP H
EJECT
* TEST BYTE FOR LOWER CASE, CONVERT TO UPPER
      CPI 1730
      JNC RC.STRD
      CPI 1410
      JC RC.STRD
      SUI 400
* STORE BYTE IN (D)
RC.STRD MOV D,A
* TEST BYTE FOR CARRIAGE RETURN
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        CPI      150
        JZ       RC.DONE
* TEST BYTE FOR BACKSPACE
        CPI      100
        JNZ      RC.TSTB
* IGNORE BYTE IF NO DATA TO BACKSPACE
        MOV      A,C
        ANA      A
        JZ       RC.CKQ
* BACKSPACE
        INR      B
        DCR      C
        DCX      H
        JMP      RC.SKIP
* IGNORE BYTE IF CONTROL CHARACTER
RC.TSTB  CPI      / /
        JC       RC.CKQ
* IGNORE BYTE IF END OF TARGET
        MOV      A,B
        ANA      A
        JZ       RC.CKQ
* MOVE BYTE TO THE TARGET
        MOV      M,D
        INX      H
        DCR      B
        INR      C
RC.SKIP  CALL     RC.ECHO
        JMP      RC.CKQ
* MOVE COMPLETE, LINE FEED AND RETURN
RC.DONE  CALL     RC.ECHO
        MVI      D,120
* ECHO BYTE TO CONSOLE, ENTRY: (D) = BYTE
RC.ECHO  IN       RC.CSST
        RAR
        JNC      RC.ECHO
        MOV      A,D
        OUT      RC.CSDT
        RET
* EL-RC.
* SL-RT.
        STL     READ TAPE RECORD
        VERSION 00.00 780227'
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

EJECT

```
*                                     BY: RON TURNER
RT.TPC    EQU    3710
RT.ABT    EQU    2244A
RT.SRS    EQU    2265A
RT.2BYT   EQU    2325A
RT.1BYT   EQU    2331A
RT.OKSM   EQU    40027A
RT.ERRX   EQU    40031A
RT.CMSG   DB     'TAPE PARITY ERROR',150,120
RT.IMG    DB     'IGNORE? (Y/N) '
RT.RPLY   DS     1
    SPACE 2
* SAVE TARGET LENGTH, TARGET ADDRESS
RT.RTN    PUSH    B
          PUSH    H
* SET UP TAPE CANCEL ADDRESS
          LXI     H,RT.ABT
          SHLD    RT.ERRX
* (HL) = DATA LENGTH, (D) = FORMAT, (E) = SEQ#
          CALL    RT.SRS
          MOV     L,A
* MOVE FORMAT, SEQ# AND DATA LENGTH TO TARGET
          POP     B
          MOV     A,D
          STAX    B
          INX     B
          MOV     A,E
          STAX    B
          INX     B
          MOV     A,H
          STAX    B
          INX     B
          MOV     A,L
          STAX    B
          INX     B
* IF MEMORY IMAGE, ADD 4 TO DATA LENGTH
          MOV     A,D
          ANI     1770
          CPI     1
          JNZ     RT.NOAD
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        INX     H
        INX     H
        INX     H
        INX     H
* (DE) = DATA LENGTH
RT.NOAD  XCHG
        EJECT
* (HL) -> TARGET, (BC) = TARGET LENGTH
        PUSH    B
        POP     H
        POP     B
        DCX     B
        DCX     B
        DCX     B
        DCX     B
* TEST FOR END OF DATA
RT.TEST  MOV     A,D
        ORA     E
        JZ      RT.END
* READ A BYTE OF DATA
        CALL    RT.1BYT
        MOV     M,A
        DCX     D
* TEST FOR END OF TARGET
        MOV     A,B
        ORA     C
        JZ      RT.TEST
        DCX     B
        INX     H
        JMP     RT.TEST
* CALCULATE THE CHECKSUM
RT.END   CALL    RT.2BYT
* TURN OFF THE TAPE
        XRA     A
        OUT     RT.TPC
* TEST THE CHECKSUM
        LHLD    RT.CKSM
        MOV     A,H
        ORA     L
        RZ
* HAVE CHECKSUM ERROR, ALERT OPERATOR
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        LXI    D,RT.CMSG
        MVI    B,19
        CALL   WC.RTN
        CALL   AO.RTN
* REQUEST OPERATOR RESPONSE
        LXI    D,RT.IMG
        MVI    B,14
        CALL   WC.RTN
        LXI    H,RT.RPLY
        MVI    B,1
        CALL   RC.RTN
        LDA    RT.RPLY
        CPI    'Y'
        RZ
* NEGATIVE RESPONSE, CTL-C INTERRUPT
        LDA    CI.CTL
        ORI    2400
        STA    CI.CTL
        LALD   CI.UINT
        PCHL
* EL-RT.
* SL-SL.
        STL    'SHIFT BITS LEFT'          VERSION 00.00 780322'
        EJECT
* TURN OFF CARRY                                BY: RON TURNER
SL.RTN   ANA    A
SL.SHFT  MOV    A,M
        RAL
        MOV    M,A
        INX    H
        DCR    B
        JNZ    SL.SHFT
        RET
* EL-SL.
* SL-WC.
        STL    'WRITE CONSOLE'             VERSION 00.00 780302'
        EJECT
*
                                BY: RON TURNER
WC.CSDT  EQU    3720             CONSOLE DATA
WC.CSST  EQU    3730             CONSOLE STATUS
        SPACE 2
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

\* TEST FOR CTL-S

WC.RTN LDA CI.CTL  
RAR  
JC WC.RTN

\* TEST FOR CTL-O

RAR  
JC WC.SKIP

\* TEST IF DEVICE READY

WC.WAIT IN WC.CSST  
RAR  
JNC WC.WAIT

\* WRITE A BYTE OF DATA

LDAX D  
OUT WC.CSDT  
WC.SKIP INX D  
DCR B  
JNZ WC.RTN  
RET

\* EL-WC.

\* SL-WT.

STL 'WRITE TAPE RECORD  
EJECT

VERSION 00.00 780227'

\*

BY: RON TURNER

WT.TPC	EQU	3710	TAPE CONTROL
WT.ABT	EQU	2244A	TAPE ERROR ROUTINE
WT.2BYT	EQU	3017A	WRITE 2 BYTES
WT.1BYT	EQU	3024A	WRITE 1 BYTE
WT.CKSM	EQU	40027A	TAPE CHECKSUM
WT.ERRX	EQU	40031A	-> TAPE ERROR ROUTINE

SPACE 2

\* SET UP TAPE CANCEL ADDRESS

WT.RTN LXI H,WT.ABT  
SHLD WT.ERRX

\* SET UP TAPE CONTROL

MVI A,1  
OUT WT.TPC

\* WRITE 32 SYNC CHARACTERS

MVI A,22  
MVI H,32  
WT.SYNC CALL WT.1BYT  
DCR H

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        JNZ     WT.SYNC
* WRITE STX CHARACTER
        MUI     A,2
        CALL    WT.1BYT
EJECT
* ZERO CHECKSUM
        MOV     L,H
        SHLD    WT.CKSM
* WRITE FORMAT AND SEQUENCE NUMBER
        MUI     B,2
WT.FMSQ LDAX    D
        CALL    WT.1BYT
        INX     D
        DCR     B
        JNZ     WT.FMSQ
* WRITE DATA LENGTH, (HL) = DATA LENGTH
        LDAX    D
        MOV     H,A
        CALL    WT.1BYT
        INX     D
        LDAX    D
        MOV     L,A
        CALL    WT.1BYT
        INX     D
* TEST IF ALL DATA WRITTEN
WT.TEST MOV     A,H
        ORA     L
        JZ      WT.CSUM
* WRITE A BYTE OF DATA
        LDAX    D
        CALL    WT.1BYT
        INX     D
        DCX     H
        JMP     WT.TEST
* WRITE THE CHECKSUM; TURN OFF THE TAPE
WT.CSUM LHLD    WT.CKSM
        CALL    WT.2BYT
        CALL    WT.2BYT
        XRA     A
        OUT     WT.TPC
        RET
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

\* EL-WT.

\* SL-XH.

STL 'EXPAND TO HEX ASCII' VERSION 00.00 780301'

EJECT

XH.RTN LDAX D BY: RON TURNER

ANI 3600

RRC

RRC

RRC

RRC

CALL XH.CURT

LDAX D

ANI 170

CALL XH.CURT

INX D

RET

XH.CURT ORI 600

CPI 720

JC XH.LOWR

ADI 70

XH.LOWR MOV M,A

INX H

RET

\* EL-XH.

\* SL-XO.

STL 'EXPAND TO OCTAL ASCII' VERSION 00.00 780301'

EJECT

XO.RTN LDAX D BY: RON TURNER

ANI 3000

RLC

RLC

CALL XO.CURT

LDAX D

ANI 700

RRC

RRC

RRC

CALL XO.CURT

LDAX D

ANI 70

CALL XO.CURT



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
      INX      D
      RET
XO.CURT  ORI      600
      MOV      M,A
      INX      H
      RET
```

\* EL-XO.

\* SL-ZM.

STL 'ZERO MEMORY

VERSION 00.00 780217'

EJECT

```
ZM.RTN  MVI      M,0
      INX      H
      DCR      B
      JNZ      ZM.RTN
      RET
```

BY: RON TURNER

\* EL-ZM.

\* SS-TAPEM

STL 'STORAGE AREAS

VERSION 00.00 780503'

EJECT

```
TABLE  EQU      *
      DB      'COPY',0
      DW      COPY
      DB      'READ',0
      DW      READ
      DB      'LISTQ',0
      DW      LISTQ
      DB      'LISTX',0
      DW      LISTX
      DB      'WRITE',0
      DW      WRITE
      DB      '?',0
      DW      HELP
      DB      'LABEL',0
      DW      LABEL
      DB      'DUMMY',0
      DW      DUMMY
      DB      'EOF',0
      DW      EOF
      DB      'SEQ',0
      DW      SEQ
      DB      0
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

TPLNTH      DW      0
LNFEED      DB      120
PROMPT      DB      'COMMAND? '
DONE        DB      'DONE, NEXT '
FMTMSG1     DB      'FORMAT= '
FMTMSG2     EQU     *-1
            DB      ', '
RECMSG      DB      'REC#= '
NOLBL       DB      'NO LABEL'
FEED        DB      150,120
LBLEQ       DB      'LABEL= '
EOFMSG      DB      '=EOF',150,120
CPYMSG1     DB      150,120,'ERROR: BUFFER FORMAT= '
ASFMTN2     EQU     *-1
            DB      ', REC#= '
ASSEQN2     EQU     *-1
            DB      150,120
CPYSIZ1     EQU     *-CPYMSG1
SEMICLN     DB      ';'
EJECT
HLPMSG1     DB      'READ - READ RECORD INTO BUFFER'
            DB      150,120
            DB      'WRITE - WRITE RECORD FROM BUFFER'
            DB      150,120
            DB      'COPY N - COPY "N" # OF FILES'
            DB      150,120
            DB      'LABEL CCC - WRITE LABEL CCC'
            DB      150,120
            DB      'DUMMY N - WRITE DUMMY SEQ# N'
            DB      150,120
HLPsiz1     EQU     *-HLPMSG1
HLPMSG2     DB      'EOF FFF SSS - WRITE EOF FMT SEQ#'
            DB      150,120
            DB      'SEQ N - CHANGE SEQ# TO N'
            DB      150,120
            DB      'LISTO - OCTAL DUMP OF BUFFER'
            DB      150,120
            DB      'LISTX - HEX DUMP OF BUFFER'
            DB      150,120,120
HLPsiz2     EQU     *-HLPMSG2
BINFMTN     DS      1

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```
BINSEQN    DS      1
RECCNT     DS      2
TPCNT      DS      2
BINARY2    DS      2
CNSSEQ01   DS      3
CNSSEQ02   EQU     *-1
INPUT      DS      20
CNSLBL     DS      63
CNSOUT     EQU     *
           DS      4
CNSLSB1    EQU     *-1
           DB      /-/'
           DS      4
CNSLSB2    EQU     *-1
           DB      /-/'
CNSCHR     DS      18
CNSEXP     DS      51
           ERRNZ   *-CNSOUT-80
DMREC      EQU     *
DMFMT      DB      0
DMSEQ      DB      0
DMSIZ      DW      0
DMDATA     DS      14
EOFREC     DB      0,0,0,0
TPAREA     EQU     *
TPFMT      DB      0
TPSEQ      DB      0
TPSIZE     DW      0
```

STL 'PROGRAM INITIALIZATION VERSION 00.00 780503'

EJECT

TPDATA CALL PS.RTN

\* ESTABLISH USER INTERRUPT ROUTINE

LXI H,PI.RTN

SHLD CI.UINT

\* CALCULATE MAXIMUM RECORD LENGTH

LXI H,65435-TPAREA

DAD SP

SHLD TPLNTH

LXI D,TPLNTH

LXI H,RECSZ2

MVI B,2

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: TAPE MANAGEMENT - TURNER < CONT'D >

```

        MUI      C,5
        CALL     BC.RTN
* DISPLAY PROGRAM TITLE
        LXI      D,TITLE
        MUI      B,TITLE$Z
        CALL     WC.RTN
* RE-ESTABLISH COLD START
        LXI      H,PS.RTN
        SHLD     COLD+1
        RET
TITLE    DB      'TAPE MANAGEMENT FOR H-8'
        DB      150,120
        DB      'BY: RON TURNER  STANFORD IL'
        DB      150,120
        DB      'VERSION 00.00 EFFECTIVE 780601'
        DB      150,120,120
        DB      'MAXIMUM PERMISSABLE RECORD '
        DB      'LENGTH IS '
RECSZ2   EQU     *-1
        DB      150,120,120
        DB      'ENTER A QUESTION MARK (?) '
        DB      'FOR COMMAND LIST'
        DB      150,120,120
TITLE$Z  EQU     *-TITLE
        END      START
* ES-TAPEM
```

.....



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN

```

        TITLE      'PGM1 - DUMP/DISASSEMBLER PROGRAM'
        STL        'INITIALIZATION/INPUT CODE'
** DUMP/DISASSEMBLER PROGRAM FOR H8 (8080) SYSTEM
** DERIVED FROM BASIC PROGRAM PUBLISHED IN REMARK, ISSUE 2, 1978
** COPYRIGHT 1978, J H SHIFFRIN
PGM1    ORG        150000H
        LXI        H,CTLC      GET ADDRESS OF CTL-C ROUTINE
        SHLD       $CSIC      AND STORE WITHIN CONSOLE DRIVER
        CALL       $PRSC      SET UP CONSOLE
START   LXI        SP,STACK    INITIALIZE STACK POINTER
        LXI        H,PRMPT     POINT TO PROMPT MSG
        MUI        B,LPRMPT     LENGTH FOR LOOP
        CALL       PRINT      PRINT THE PROMPT MESSAGE
* PROMPT MESSAGE ISSUED, NOW READ MEMORY LIMITS
        LXI        H,LOW       INITIALIZE
RDLP1   LXI        D,0
        MUI        C,77        FLAG TO BE USED TO FIND FIRST DIGIT
        MUI        B,6
RDLP    CALL       $RCHAR      READ ONE CHAR
        CPI        CR          CR
        JZ         RDEND      EXIT IF CR
        CPI        COMMA      COMMA
        JNZ        SKIP      SKIP
* END OF FIRST NUMBER, STORE IT AND DO HOUSEKEEPING
        STA        TEMP        TEMPORARY SAVE
        MUI        A,#LOW      GET LOW HALF ADDRESS OF LOW
        CMP        L          DOES HL STILL POINT TO LOW
        JNZ        ERR        NO, THIS IS NOT THE FIRST COMMA, REJ
ECT IT  LDA        TEMP        RETRIEVE CHARACTER
        CALL       $WCHAR      ECHO COMMA
        CALL       STORE      GO STORE DE IN 'LOW'
        INX        H          POINT TO 'HIGH'
        JMP        RDLP1      GO GET NEXT NUMBER
SKIP    STA        TEMP        TEMPORARY SAVE
        MOV        A,B
        ORA        A
        JZ         ERR        TOO MANY DIGITS
        LDA        TEMP
        CPI        600
        JM         ERR        NUMBER TOO SMALL

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

        CPI        700
        JP         ERR        NUMBER TOO LARGE
        CALL      $WCHAR      ECHO NUMBER
        XCHG
        XCHG        SWAP DE AND HL
*** THE FOLLOWING CODE SAVES THE FIRST INPUT DIGIT.  IF IT IS
*** A 2 OR 3 AND 6 DIGITS ARE INPUT, THE HIGH ORDER BIT OF D
*** WILL BE TURNED ON BEFORE D IS STORED.
        PUSH      PSW        SAVE INPUT CHARACTER
        MOV       A,C
        CPI       77        SEE IF IT IS THE FIRST DIGIT
        JNZ       NOT1ST
        POP       PSW
        MOV       C,A        SAVE FIRST DIGIT IN C
        JMP       FIRST
NOT1ST  POP       PSW
FIRST   EQU       *        END OF SAVE CODE
        DAD       H
        DAD       H
        DAD       H        (HL)*8
        XCHG        SWAP BACK
        SUI       600        REMOVE BIAS FROM NUMBER
        ADD       E
        MOV       E,A        ADD DIGIT TO E
        DCR       B        DIGIT COUNTER
        JMP       RDLP        GO GET NEXT DIGIT (OR DELIMITER)
RDEND   CALL      STORE        MOVE NUMBER FROM DE TO HIGH
        MVI       A,CR
        CALL      $WCHAR
        MVI       A,LF
        CALL      $WCHAR      PRINT CR,LF
*** IF HIGH = 000.000. THEN HALT.
        LXI       H,HIGH
        MOV       A,M
        INX       H        POINT TO HIGH+1
        CMP       M        COMPARE THE TWO BYTES OF HIGH
        JNZ       CHOICE    IF NOT EQUAL, THEN HIGH <> 0.0
        CPI       0
        JNZ       CHOICE    THEY ARE EQUAL, BUT NOT 0
        HLT
        JMP       START      SET UP FOR POSSIBLE RESTART
CHOICE  LXI       H,PRMPT2    POINT TO SECOND PROMPT MESSAGE

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

MVI      B,LPRMP2  LENGTH OF MESSAGE
CALL     PRINT     ASK FOR DUMP/DISAS OPTION
CALL     $RCHAR    GET ANSWER
CALL     $WCHAR    ECHO IT
SUI      600       NORMALIZE TO AN OCTAL DIGIT
JZ       DUMP      IF INPUT=0, MEMORY DUMP DESIRED
DCR      A
JZ       DISAS     IF INPUT=1, DISASSEMBLY DESIRED
JMP      CHOICE    BAD INPUT, TRY AGAIN
STL      'DUMP CODE'
EJECT

```

\*\* THE FOLLOWING CODE WILL PRINT A MEMORY DUMP FROM <LOW> TO <HIGH>

\*\* IN OCTAL AND ASCII, 8 BYTES PER LINE.

```

DUMP      LXI      H,LOW+1
          MOV      A,M      GET HIGH BYTE OF 'LOW'
          LXI      H,ADDRH
          CALL     CNVRT    CONVERT TO ASCII AND STORE IN LINE
          LXI      H,LOW
          MOV      A,M      GET LOW BYTE OF 'LOW'
          ANI      3700     TRUNCATE TO MULTIPLE OF 100
          MOV      M,A      AND RETURN TO LOW
          LXI      H,ADDRL
          CALL     CNVRT    CONVERT TO ASCII AND STORE IN LINE
          MVI      B,8      LOOP COUNTER
          LHALD    LOW      POINT TO MEMORY AREA TO BE DUMPED
          LXI      D,7
          DAD      D        HL NOW POINTS TO BYTE THAT GOES AT E
ND OF LINE
LP3       MOV      A,M      GET A BYTE
          PUSH     H        SAVE MEMORY POINTER
          PUSH     PSW      SAVE DATA BYTE TWICE
          PUSH     PSW
          LXI      H,DATAOC-1
          MOV      A,B      LOOP COUNTER USED AS POSITION COUNT
R         DCR      A        B-1
          RLC      C        (B-1)*2
          MOV      C,A
          RLC      C        (B-1)*4
          ADD      C        (B-1)*6

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

C          ADI          1          (B-1)*6+1 - PROPER OFFSET INTO DATA0
          ADD          L
          MOV          L,A          L NOW POINTS TO THE RIGHT SPOT IN 'L
INE'
          POP          PSW
          CALL         CNVRT        CONVERT BYTE TO THREE ASCII CHARS
          LXI          H,DATAAS-1
          MOV          A,L
          ADD          B
          MOV          L,A          POINT TO PROPER LOCATION IN DATAAS
          POP          PSW          RETRIEVE DATA BYTE
          CPI          400          IF CHAR IS
          JM           NOGOOD       NOT PRINTABLE
          CPI          1400         REPLACE IT
          JM           OK           WITH A
NOGOOD     MVI          A,560        PERIOD
OK          MOV          M,A          STORE ASCII IN DATAAS
          POP          H            RETRIEVE AND
          DCX          H            INCREMENT MEMORY POINTER
          DCR          B
          JNZ         LP3          DO IT 8 TIMES
** LINE IS COMPLETE, NOW PRINT IT.
          LXI          H,LINE        POINT TO LINE
          MVI          B,LLINE       LENGTH OF LINE
          CALL         PRINT
** NOW INCREMENT LOW, AND IF STILL LESS THAN HIGH, GO TO FORMAT A
NOTHER
** LINE. IF LOW > HIGH, RETURN TO THE BEGINNING TO GET ANOTHER S
ET
** OF MEMORY LIMITS.
          LHLD         LOW
          LXI          D,100
          DAD          D            LOW = LOW + 100
          SHLD        LOW          PUT NEW VALUE BACK IN MEMORY
          CALL         CMP          SEE IF WE'RE DONE
          JC          DUMP         NOT DONE WITH THE DUMP, LOOP
          JMP          START        DONE, GO SEE WHAT IS TO BE DONE NEXT
          STL          'DISASSEMBLER CODE'
          EJECT
** DISASSEMBLER - PRINTS MNEMONICS AND ASCII FOR ALL BYTES BETWEEN
N

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN

\*\* (LOW) AND (HIGH).

```

DISAS  LXI      H,LINE2+2  CLEAR LINE2 TO BLANKS
        MUI      B,LLINE2-2
        MUI      A,400
CLR     MOV      M,A
        INX      H          BUMP POINTERS
        DCR      B
        JNZ      CLR
        LXI      H,LINE2+2  SET UP POINTER TO USE FIRST HALF OF
LINE2
        SHLD     LIN
        MUI      A,-1      SET FLAG TO INDICATE WHICH HALF OF L
INE2 TO USE
LP4     PUSH     PSW        SAVE FLAG
        LXI      H,LOW+1
        MOV      A,M
        LHLD     LIN
        LXI      D,ADDR1
        DAD      D          POINT TO SPOT TO STORE FIRST BYTE OF
ADDRESS
        CALL     CNVRT      CONVERT TO ASCII AND STORE IN LINE2
        MUI      A,560
        LHLD     LIN
        LXI      D,PERD
        DAD      D
        MOV      M,A        STORE '.'/ IN LINE2 BETWEEN ADRESS BY
TES
        LXI      H,LOW
        MOV      A,M        GET LOW BYTE OF 'LOW'
        LHLD     LIN
        LXI      D,ADDR2
        DAD      D
        CALL     CNVRT      CONVERT AND STORE IN LINE2 AT PROPER
SPOT
        CALL     GET        GET A INSTRUCTION (OR DATA) BYTE
        LHLD     LIN
        LXI      D,ASCII
        DAD      D
        CALL     PUT        PUT ITS ASCII VALUE INTO LINE2
        LXI      H,TABLE1   POINT TO INSTRUCTION LENGTH TABLE
        MOV      E,A        ADD

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

        MUI      D,0          IN PROPER
        DAD      D            OFFSET FROM TABLE
        MOV      C,M          GET INSTRUCTION LENGTH
* A HAS INSTRUCTION BYTE - MUST MULTIPLY BY 8 TO GET ADDRESS OFFSET
ET
* INTO TABLE2
        MUI      B,3
        LXI      D,0
RLP      RAL              A*2, MSB INTO CARRY
        MOV      E,A
        MOV      A,D
        RAL              ROTATE CARRY INTO E
        MOV      D,A
        MOV      A,E
        DCR      B
        JNZ      RLP        DO IT THREE TIMES
        LXI      H,TABLE2    POINT TO TABLE2 (DE HAS OFFSET)
        DAD      D
        LXI      D,OPCODE
        PUSH     H
        LHLD     LIN
        DAD      D
        XCHG              DE NOW POINTS TO SPOT TO STORE OPCODE
E
        POP      H          HL NOW POINTS TO SPOT IN TABLE2 TO GET
ET OPCODE
* MOVE 8 BYTES FROM TABLE2 TO LINE2 (AT LOCATION OPCODE)
        MUI      B,8
MOULP    MOV      A,M          GET BYTE FROM TABLE
        XCHG              HL POINTS TO LINE2
        MOV      M,A          STORE BYTE
        XCHG              HL POINTS TO TABLE2
        INX      D            INCREMENT POINTERS
        INX      H
        DCR      B
        JNZ      MOULP
* IF THIS IS A MULTIPLE BYTE INSTRUCTION, HANDLE THE OTHER BYTES.
        MOV      A,C          RETRIEVE LENGTH
        ORA      A            CHECK FOR ZERO
        JZ       NOMORE      AND JUMP IF ZERO
        LHLD     LIN

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

        LXI        D,ASCII+1
        DAD        D
        SHLD       ASC      LOCATION FOR ASCII OF NEXT BYTE
        LHL        LIN
        LXI        D,BYTE2
        DAD        D
        SHLD       BYTE     LOCATION FOR OCTAL OF NEXT BYTE
BYTE23  CALL       GET      GET BYTE
        LHL        ASC      POINT TO SPOT IN LINE
        CALL       PUT      MOVE BYTE TO LINE (ASCII)
        LHL        BYTE
        CALL       CNVRT    MOVE OCTAL TO LINE2
        LHL        ASC      BUMP POINTER FOR NEXT BYTE
        LXI        D,1
        DAD        D
        SHLD       ASC
        LHL        BYTE     BUMP OTHER POINTER FOR NEXT BYTE
        LXI        D,4
        DAD        D
        SHLD       BYTE
        DCR        C
        JNZ        BYTE23   JUMP BACK IF THERE IS A THIRD BYTE
* DONE WITH THIS HALF OF LINE2.  IF IT'S THE FIRST HALF, INCREMENT
T POINTERS
* AND GO FILL IN THE SECOND HALF.  IF IT'S THE SECOND HALF, PRINT
THE LINE
* AND CHECK TO SEE IF ALL MEMORY HAS BEEN HANDLED.  NOTE: DISAS W
ILL ALWAYS
* PRINT FULL LINES, AND MAY PRINT ONE MORE INSTRUCTION THAN REQUE
STED.
NOMORE  LHL        LIN      INCREMENT LINE2 POINTER
        LXI        D,40
        DAD        D
        SHLD       LIN
        POP        PSW      RETRIEVE LINE FLAG AND SEE WHERE WE
ARE
        INR        A        A WILL NOW BE 0 IF LINE2 IS HALF FUL
L
        JZ         LP4      IF SO GO DO IT AGAIN.
* LINE IS FULL, PRINT IT AND SEE IF IT IS TIME TO QUIT
        LXI        H,LINE2
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

MUI      B,LLINE2
CALL     PRINT
LHLD     LOW
CALL     CMP      SEE IF WE'RE DONE
JC       DISAS    NO, LOOP
JMP      START    YES, GO SEE WHAT IS TO BE DONE NEXT
STL      'SUBROUTINES'
EJECT

```

\*\* PRINT - SUBROUTINE TO PRINT ASCII TEXT.

\*\* H,L POINTS TO TEXT, B HAS CHARACTER COUNT.

```

PRINT    MOV      A,M      GET A CHARACTER
          CALL     $WCHAR   PRINT IT
          INX      H
          DCR      B
          JNZ      PRINT    LOOP
          RET

```

\*\* CHURT - SUBROUTINE TO CONVERT A BYTE TO THREE ASCII CHARACTERS

\*\* A CONTAINS BYTE, H,L POINTS TO DESTINATION FOR ASCII.

```

CHURT    EQU      *
          PUSH     B
          MOV      B,A      COPY OF BYTE
          ANI      3000     ISOLATE LEFT 2 BITS
          RLC      MOVE TO OTHER
          RLC      END OF A
          ADI      600      MAKE IT ASCII
          MOV      M,A      STORE FIRST DIGIT
          INX      H
          MOV      A,B      RETRIEVE BYTE
          ANI      0700     GET MIDDLE 3 BITS
          RRC      MOVE TO
          RRC      RIGHT END
          RRC      OF A
          ADI      600      MAKE IT ASCII
          MOV      M,A      STORE MIDDLE DIGIT
          INX      H
          MOV      A,B
          ANI      0070     GET RIGHT 3 BITS
          ADI      600      MAKE IT ASCII
          MOV      M,A      STORE LAST DIGIT
          POP      B

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

                RET
** STORE - SUBROUTINE TO STORE D,E IN MEMORY (IN REVERSE ORDER)
STORE    MOV     M,E      STORE E AT LOC FOR LSB
        INX     H
        MOV     A,D      D NEEDS TO BE SHIFTED BACK ONE BIT R
IGHT
        RAR
        MOV     D,A
        DCR     B      IF B GOES NEGATIVE, 6 DIGITS WERE IN
PUT
        JP      NOADD    JUMP AROUND IF LESS THAN 6 DIGITS IN
PUT
        MOV     A,C
        ANI     2      ISOLATE PROPER BIT
        RRC
        RRC      MOVE BIT TO MSB POSITION
        ADD     D
        MOV     D,A      RETURN BYTE TO THE D REGISTER
NOADD    MOV     M,D      STORE CONTENTS OF D IN MEMORY
        RET
** GET - SUBROUTINE TO GET A BYTE AT (LOW), THEN INCREMENT (LOW).
** USES DE, RETURNS BYTE IN A
GET      LHLD    LOW     POINT TO DATA BYTE
        MOV     A,M
        LXI     D,1
        DAD     D      INCREMENT POINTER
        SHLD    LOW     SAVE POINTER
        RET
** PUT - SUBROUTINE TO PUT ASCII (OR '.' IF ASCII IS NOT PRINTABL
E)
** INTO LINE2 AT LOC SPECIFIED BY HL.
PUT      PUSH    PSW      SAVE ORIGINAL BYTE
        CPI     400      IS IT < 400 (SPACE)
        JM      NEEDPR
        CPI     1400     IS IT > 1370 (.)
        JM      POK
NEEDPR   MVI     A,560    NON-PRINTABLE CHAR, USE '.'
POK      MOV     M,A
        POP     PSW      RETRIEVE ORIGINAL BYTE
        RET
** CMP - SUBROUTINE TO COMPARE THE CURRENT VALUES OF (LOW) AND (H
IGH)

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN

\*\* AT ENTRY, HL HAS <LOW>. AT EXIT, THE CARRY FLAG IS SET IF <LOW>  
> < <HIGH>

```

CMP      XCHG      MOVE <LOW> TO DE
          LALD      HIGH    GET <HIGH> IN HL
          MOV       A,D     COPY MSB OF <LOW>
          CMA       COMPLEMENT IT
          MOV       D,A     AND RETURN TO D
          MOV       A,E     COPY LSB OF <LOW>
          CMA       COMPLEMENT IT
          MOV       E,A     AND RETURN TO E
          DAD       D       EQUIVALENT OF <HIGH> - <LOW>
          RET

```

\*\* ERR-ROUTINE TO SEND BELL TO CONSOLE AND RETURN TO READ ANOTHER  
CHARACTER

\*\* ERR IS CALLED WHEN A NON-DIGIT CHARACTER IS INPUT, OR WHEN NO  
RE THAN 6

\*\* DIGITS ARE INPUT FOR AN OCTAL VALUE.

```

ERR      MVI       A,7
          CALL      $WCHAR    RING BELL
          JMP       RDLF     TRY AGAIN

```

\*\* CTLC - ROUTINE TO HANDLE CONTROL-C INPUT. IT SIMPLY REENABLES  
\*\* INTERRUPTS AND RESTARTS THE PROGRAM.

```

CTLC     EI        ENABLE INTERRUPTS
          JMP       START    RESTART PROGRAM
          STL       'DATA AND CONSTANTS'
          EJECT

```

\*\* DATA AND CONSTANTS \*\*

```

$CSIC    EQU       040250A    SPOT TO STORE ADDRESS OF CTL-C ROUTI
NE
$PRSC    EQU       040152A    CONSOLE SETUP ADDRESS
$RCHAR   EQU       040144A    READ CHARACTER ROUTINE
$WCHAR   EQU       040147A    WRITE CHARACTER ROUTINE
LOW      DS        2         LOW MEMORY ADDRESS
HIGH     DS        2         HIGH MEMORY ADDRESS
CR       EQU       150
LF       EQU       120
COMMA    EQU       540
PRMPT    DB        CR,LF,'ENTER LIMITS IN OCTAL:'
LPRMPT   EQU       *-PRMPT
PRMPT2   DB        CR,LF,'ENTER 0 FOR MEMORY DUMP, 1 FOR DISASSEM
BLV:'
LPRMPT2  EQU       *-PRMPT2

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

TEMP      DS      1      TEMPORARY STORAGE
LINE      DB      CR,LF
ADDRH     DS      3
          DB      /
ADDRL     DS      3
          DB      /
DATAOC    DB      /
          DB      /*
DATAAS    DB      /
          DB      /*
LLINE     EQU     *-LINE
LINE2     DB      CR,LF      PRINT LINE FOR DISASSEMBLER
          DS      75
LLINE2    EQU     *-LINE2
* THE FOLLOWING EQUATES ARE USED TO POSITION ITEMS WITHIN LINE2
ADDR1     EQU     0
PERD      EQU     3
ADDR2     EQU     4
OPCODE    EQU     10
BYTE2     EQU     22
BYTE3     EQU     26
ASCII     EQU     32
LIN       DS      2      HOLDS ADDRESS OF AREA IN LINE2 TO BE
          USED
ASC       DS      2      HOLDS ADDRESS FOR ASCII OF BYTE 'N' (
N=1,2,3)
BYTE      DS      2      HOLDS ADDRESS FOR OCTAL OF BYTE 'N' (
N=2,3)
          STL      'TABLE OF INSTRUCTION LENGTHS'
          EJECT
* (TABLE1+I) = LENGTH-1 OF INSTRUCTION I (0<=I<=255)
TABLE1    DB      0,2,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0 (000 - 017)
          DB      0,2,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0 (020 - 037)
          DB      0,2,2,0,0,0,0,1,0,0,0,0,2,0,0,0,1,0 (040 - 057)
          DB      0,2,2,0,0,0,0,1,0,0,0,0,2,0,0,0,1,0 (060 - 077)
          DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (100 - 117)
          DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (120 - 137)
          DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (140 - 157)
          DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (160 - 177)
          DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (200 - 217)

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN < CONT'D >

```

      DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (220 - 237)
      DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (240 - 257)
      DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 (260 - 277)
      DB      0,0,2,2,2,0,1,0,0,0,2,0,2,2,1,0 (300 - 317)
      DB      0,0,2,1,2,0,1,0,0,0,2,1,2,0,1,0 (320 - 337)
      DB      0,0,2,0,2,0,1,0,0,0,2,0,2,0,1,0 (340 - 357)
      DB      0,0,2,0,2,0,1,0,0,0,2,0,2,0,1,0 (360 - 377)
      STL      'ASCII MNEMONICS FOR INSTRUCTIONS'
      EJECT

* TABLE2 - 256 8 BYTE ENTRIES, ONE FOR EACH INSTRUCTION
TABLE2 DB 'NOP      LXI B      STAX B      INX B      INR B      DCR B      MUI
B      RLC      '
      DB 'NOP      DAD B      LDAX B      DCX B      INR C      DCR C      MUI
C      RRC      '
      DB 'NOP      LXI D      STAX D      INX D      INR D      DCR D      MUI
D      RAL      '
      DB 'NOP      DAD D      LDAX D      DCX D      INR E      DCR E      MUI
E      RAR      '
      DB 'NOP      LXI H      SHLD      INX H      INR H      DCR H      MUI
H      DAA      '
      DB 'NOP      DAD H      LHLD      DCX H      INR L      DCR L      MUI
L      CMA      '
      DB 'NOP      LXI SP     STA      INX SP     INR M      DCR M      MUI
M      STC      '
      DB 'NOP      DAD SP     LDA      DCX SP     INR A      DCR A      MUI
A      CMC      '
      DB 'MOV B,B MOV B,C MOV B,D MOV B,E MOV B,H MOV B,L MOV
B,M MOV B,A '
      DB 'MOV C,B MOV C,C MOV C,D MOV C,E MOV C,H MOV C,L MOV
C,M MOV C,A '
      DB 'MOV D,B MOV D,C MOV D,D MOV D,E MOV D,H MOV D,L MOV
D,M MOV D,A '
      DB 'MOV E,B MOV E,C MOV E,D MOV E,E MOV E,H MOV E,L MOV
E,M MOV E,A '
      DB 'MOV H,B MOV H,C MOV H,D MOV H,E MOV H,H MOV H,L MOV
H,M MOV H,A '
      DB 'MOV L,B MOV L,C MOV L,D MOV L,E MOV L,H MOV L,L MOV
L,M MOV L,A '
      DB 'MOV M,B MOV M,C MOV M,D MOV M,E MOV M,H MOV M,L HLT
      MOV M,A '
      DB 'MOV A,B MOV A,C MOV A,D MOV A,E MOV A,H MOV A,L MOV
A,M MOV A,A '

```

PROGRAM NAME: 8080 DISASSEMBLER - SHIFFRIN

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	00
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

<H<U<G> <S<O<F<T<W<A<R<E> <U<O<L> II

PROGRAM NAME: CCS SUPPRESS - MORGAN

```

***      TITLE 'COMMAND COMPLETION SUPPRESSING CONSOLE DRIVER'
***      COMMAND COMPLETION SUPPRESSING CONSOLE DRIVER
*
*      BY MICHAEL WILLIAM MORGAN, AUGUST 1978.
*
***      THIS CONSOLE DRIVER WILL SUPPRESS COMMAND
*      COMPLETION BY ANY PROGRAM WHICH USES THE H8
*      CONSOLE DRIVER.  THE PACKAGE PHYSICALLY
*      REPLACES THE H8 CONSOLE DRIVER IN MEMORY,
*      REQUIRING NO ADDITIONAL MEMORY, AND IS
*      COMPATIBLE WITH ANY PROGRAM WHICH USES THE
*      H8 CONSOLE DRIVER IN A NORMAL MANNER.
*
*
***      MUCH OF THIS DRIVER IS A COPY OR REVISION OF
*      THE H8 CONSOLE DRIVER, WHICH IS COPYRIGHTED
*      BY HEATH COMPANY, BENTON HARBOR, MI.
*
*
***      CONSTANTS
*
*      MACHINE INSTRUCTIONS:
MI.MVIA EQU    0760      MVI A.
MI.SUI  EQU    3260      SUI.
MI.JMP  EQU    3030      JMP.
*
*      ASCII CHARACTERS:
BELL    EQU    70        BELL.
*
*      BITS:
ECHOED  EQU    2000      $CHAR HAS BEEN ECHOED.
*
*      LOCATIONS:
UIVEC   EQU    40037A     USER INTERRUPT VECTOR.
INTEXT  EQU    00175A     INTERRUPT EXIT.
*
*
***      USER PROGRAM ENTRY POINTS
*
*      LOCATIONS 40100A THROUGH 40105A ARE USED IN
*      THE SAME MANNER AS BY THE H8 CONSOLE DRIVER.
*

```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CCS SUPPRESS - MORGAN < CONT'D >

```
ENTRY    EQU    40100A
*
*
**      PORT ROUTINES
*
*      LOCATIONS 40106A THROUGH 40143A ARE USED IN
*      THE SAME MANNER AS BY THE H8 CONSOLE DRIVER.
*
$CDIN    EQU    40106A    CONSOLE DATA IN.
$CDOUT    EQU    40111A    CONSOLE DATA OUT.
$CISI    EQU    40114A    CONSOLE INPUT STATUS IN.
$CISO    EQU    40117A    CONSOLE INPUT STATUS OUT.
$COSI    EQU    40122A    CONSOLE OUTPUT STATUS IN.
$TSOUT    EQU    40141A    TAPE STATUS OUT.
*
*
*      ORG    40144A
*
*
**      REMOTE ENTRY POINTS
*
*      LOCATIONS 40144A THROUGH 40154A ARE USED IN
*      THE SAME MANNER AS BY THE H8 CONSOLE DRIVER.
*
$RCHAR    JMP    $RCHAR.    READ ONE CHARACTER.
$WCHAR    JMP    $WCHAR.    WRITE ONE CHARACTER.
$PRSCl    JMP    $PRSCl.    PRESET CONSOLE UART.
*
*
**      $INBUF - TYPE-AHEAD BUFFER
*
*      LOCATIONS 40155A THROUGH 40205A ARE USED IN
*      THE SAME MANNER AS BY THE H8 CONSOLE DRIVER.
*
$INBUF    DB      0          BUFFER COUNT.
          DS      24          THE BUFFER.
*
$INBUFL    EQU    *-$INBUF-1    MAXIMUM BUFFER COUNT.
*
P.INBUF    EQU    $INBUF/1000A    $INBUF PAGE.
ERRNZ    */1000A-P.INBUF
```







<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CCS SUPPRESS - MORGAN < CONT'D >

```
*                                     $UPPER IS HARMLESS.
*
*
**      $UPPER - CONVERT TO UPPER CASE
*
*      ENTRY - (A) = CHARACTER.
*      EXIT  - (A) = CONVERTED CHARACTER.
*      USES  - F.
*
*                                     IF IT'S A LOWER CASE LETTER,...
$UPPER  CPI    1410
        RC
        CPI    1730
        RNC
*
*                                     CONVERT TO UPPER CASE.
        DB     MI.SUI
        ERNZ *-40307A
*
*      LOCATION 40307A IS USED IN THE SAME MANNER
*      AS BY THE H8 CONSOLE DRIVER.
*
        DS     1          ZERO TO ALLOW LOWER CASE,
                        400 FOR UPPER CASE ONLY.
*
        RET
*
*
**      $RCHAR - READ A SINGLE CHARACTER
*
*      ENTRY - NONE.
*      EXIT  - (A) = CHARACTER.
*      USES  - F.
*
$RCHAR. PUSH  H
        PUSH  D
*
*                                     WAIT UNTIL INPUT BUFFER NOT EMPTY.
        LXI   A,$INBUF
$RCHAR. MOV   A,M
        ANA   A
        JZ    $RCHAR
*
*                                     PREVENT BUFFER CHANGES BY $CSINT.
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CCS SUPPRESS - MORGAN < CONT'D >

```

DI
*          DECREMENT BUFFER COUNT.
DCR        M
**
*          RETRIEVE AND REMOVE NEXT CHARACTER.
MOV        D,A          (D) = COUNT
ADD        L
MOV        L,A          (HL) = @ LAST CHARACTER
$RCHARD    MOV        E,M
MOV        M,A
MOV        A,E
DCX        H
DCR        D
JNZ        $RCHARD      (A) = CHARACTER
*          CONVERT CHARACTER TO UPPER CASE.
CALL       $UPPER
*          SAVE IT AS LAST CHARACTER READ.
STA        $CHAR
*          ALLOW BUFFER CHANGES.
JMP        INTXIT
**
*
**
$WCHAR - WRITE A SINGLE CHARACTER
*
*          ENTRY - (A) = CHARACTER.
*          EXIT  - NONE.
*          USES  - F.
*
$WCHAR.    PUSH       D
MOV        D,A
*          IF LAST CHAR READ HAS BEEN ECHOED
*          AND IS A PRINTING CHAR,...
DB         MI.MVIA
$CHAR      DB         0          LAST CHARACTER READ.
CPI        ECHOED+' '
*          WAIT FOR MATCHING INPUT.
CNC        $MATCH
*          INDICATE CHARACTER HAS BEEN ECHOED.
ORI        ECHOED
STA        $CHAR
**
*          WAIT UNTIL OUTPUT NOT SUSPENDED.
$WCHARS    LDA        $CSLCTL

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CCS SUPPRESS - MORGAN < CONT'D >

```

    RAR
    JC    $WCHARS
*           IF OUTPUT NOT BEING DISCARDED,...
    RAR
    MOU    A,D          (A) = CHARACTER
    POP    D
    RC
*           OUTPUT THE CHARACTER.
    ERNZ $DATOUT-*
*#
*
**    $DATOUT - OUTPUT A CHARACTER
*
*    ENTRY - (A) = CHARACTER.
*    EXIT  - NONE.
*    USES  - NONE.
*
$DATOUT PUSH PSW
*           WAIT UNTIL CONSOLE READY.
$DATOUR CALL $COSI
    RAR
    JNC    $DATOUR
*           OUTPUT THE CHARACTER.
    POP    PSW
    JMP    $CDOUT
*#
*
**    $PRSC - PRESET CONSOLE DRIVERS
*
*    ENTRY - NONE.
*    EXIT  - NONE.
*    USES  - A,F,H,L.
*
*           PRESET CONSOLE UARTS.
$PRSC. MUI    A,2010
    CALL    $SOUT
    MUI    A,1000
    CALL    $SOUT
    MUI    A,1160
    CALL    $SOUT
*           CLEAR INPUT BUFFER.
```



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: CCS SUPPRESS - MORGAN < CONT'D >

```

XRA    A            (A) = 0
STA    $INBUF
*
*          SET UP CONSOLE INTERRUPT HANDLING.
MVI    A,MI.JMP
STA    UIVEC+6
LXI    H,$CSINT
SHLD   UIVEC+7
*
*          ENABLE CONSOLE INPUT.
MVI    A,270
JMP    $CISO
*
*
**      $CSINT - PROCESS A CONSOLE INTERRUPT
*
*      ENTRY - NONE.
*      EXIT  - ENABLE INTERRUPTS.
*      USES  - NONE.
*
$CSINT  PUSH    H
        PUSH    D
        PUSH    PSW
*
*          IF DATA IS AVAILABLE....
        CALL    $CISI
        ANI     2
*
*          INPUT & PROCESS THE CHARACTER.
        CNZ     $DATIN
*
        POP     PSW
        JMP     INTEXT
*
*
**      $DATIN - INPUT AND PROCESS CHARACTER
*
*      ENTRY - NONE.
*      EXIT  - NONE.
*      USES  - ALL BUT B,C.
*
*          INPUT THE CHARACTER.
$DATIN  CALL    $CDIN
        ANI     1770      (A) = CHARACTER
        RZ          DISCARD NULLS.

```

•

✱

34

生

✱

✦

✱

✱

✱

✦

生

•

✦

✱

⋮

✱

✦

 $\frac{1}{2}$ 

✦

:4:

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER

TITLE 'LABELS PROGRAM -- MAY 1978'  
STL 'CONSOLE DRIVER ROUTINES'

\*\*\*

\*

LABELS PROGRAM

\*

\*

\*

\*

PROGRAM TO READ TAPE FILE PRODUCED BY .TED.  
AND LIST ADDRESS LABELS ON THREE PART AVERY  
TYPE LABELS (3 1/2 X 15/16)

\*

\*

\*

\*

\*

PROGRAM USES OX FLOW PRINTING METHOD AND IS  
WRITTEN WITHOUT PAD CHARACTER.  
PROGRAM WAS WRITTEN TO RUN ON A DIABLO 1620-3  
TERMINAL WITH TRACTOR FEED.

\*

\*

\*

\*

PROGRAM ALSO USES MODIFIED HEATH CONSOLE  
DRIVER AND PAM ROUTINES.

\*

\*

\*

\*

\*

\*

\*

\*

\*\*\*

PROGRAM WRITTEN BY:

M. FARMER

MIKE AND KATHY DATA SERVICES

529 WEST 149TH STREET

GARDENA, CALIFORNIA 90248

DATED MAY 1978

ORG 40100A START OF USER RAM

\*\*\*

ASSEMBLY CONSTANTS

BELL

EQU 0070

.UIVEC

EQU 040037A

START

EQU 111111A DUMMY START LABEL

RESTART

EQU 041220A DUMMY RESTART LABEL

CONFIG

EQU 041201A DUMMY CONFIG LABEL

\*\*\*

8251 USART BIT DEF

\*

\*\*\*

MODE INSTRUCTION CONTROL BITS

UMI.1B

EQU 01000000B

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

UMI.HB	EQU	10000000B
UMI.2B	EQU	11000000B
UMI.PE	EQU	00100000B
UMI.PA	EQU	00010000B
UMI.L5	EQU	00000000B
UMI.L6	EQU	00000100B
UMI.L7	EQU	00001000B
UMI.L8	EQU	00001100B
UMI.1X	EQU	00000001B
UMI.16X	EQU	00000010B
UMI.64X	EQU	00000011B

\*\*\* COMMAND INSTRUCTION BITS

UCI.IR	EQU	01000000B
UCI.RO	EQU	00100000B
UCI.ER	EQU	00010000B
UCI.RE	EQU	00000100B
UCI.IE	EQU	00000010B
UCI.TE	EQU	00000001B

\*\*\* STATUS READ COMMAND BITS

USR.FE	EQU	00100000B
USR.OE	EQU	00010000B
USR.PE	EQU	00001000B
USR.TXE	EQU	00000100B
USR.RXR	EQU	00000010B
USR.TXR	EQU	00000001B

\*\*\* CONSL - SYSTEM CONSOLE AND I/O DRIVER

\*\*\* I/O PORT ADDRESSES

IP.CDP	EQU	3720
OP.CDP	EQU	3720
IP.CIS	EQU	3730
OP.CIS	EQU	3730
IP.COS	EQU	3730
OP.COS	EQU	3730

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

IP.TDP	EQU	3700
OP.TDP	EQU	3700
IP.TSP	EQU	3710
OP.TSP	EQU	3710

\*\* \$CSLCTL (CONSOLE CONTROL FLAG) BITS.

CC.HLD	EQU	01
CC.DMP	EQU	02
CC.CTLA	EQU	0100
CC.CTLB	EQU	0200
CC.CTLC	EQU	0400
CC.CTLD	EQU	1000

\*\* PROGRAM ENTRY POINTS

ENTRY	JMP	CONFIG
	JMP	RESTART

\*\* PORT ROUTINES

\$CDIN	IN	IP.CDP
\$RET	RET	

\$CDOUT	OUT	OP.CDP
	RET	

\$CISI	IN	IP.CIS
	RET	

\$CISO	OUT	OP.CIS
	RET	

\$COSI	IN	IP.COS
	RET	

\$COSO	OUT	OP.COS
	RET	

\$TDIN	IN	IP.TDP
	RET	



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

```

$TDOUT  OUT      OP.TDP
        RET

$TSIN   IN       IP.TSP
        RET

$TSOUT  OUT      OP.TSP
        RET

**      REMOTE ENTRY POINTS FOR CONSOLE DRIVER

$RCHAR  JMP      $RCHAR.  READ ONE CHARACTER
$WCHAR  JMP      $WCHAR.  WRITE ONE CHARACTER
$PRSCL  JMP      $PRSCL.  PRESET CONSOLE UART

**      DATA AND BUFFERS

$INBUF  DB        0
        DS        30

$INBUFL EQU      *-$INBUF-2

**      CONTROL CHARACTER TABLE
*
*      DB        CHAR, MASK, VALUE
*
$CSIB   DB        000, 3770, 0000    CTL-@ (00=NULL)
        DB        010, 1670, 2100    CTL-A
        DB        020, 1570, 2200    CTL-B
        DB        030, 1370, 2400    CTL-C
        DB        040, 0770, 3000    CTL-D
        DB        170, 1770, 0020    CTL-O
        DB        200, 1750, 0000    CTL-P
        DB        210, 1760, 0000    CTL-Q
        DB        230, 1760, 0010    CTL-S
        DB        1770                END OF TABLE

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

\* \$CSIC - ADD OF INTR TIME CTRL CHAR PROCESSOR

\$CSIC DW \$RET

\$CSLCTL DB 0

COLNO DB 0

\$CSLLEN DB 80

SET \$INBUF/1000A  
ERRNZ \*/1000A-.

\*\* \$RCHAR - READ SINGLE CHARACTER

\$RCHAR. PUSH H  
LXI H, \$INBUF

\$RCHAR1 MOV A, M  
ANA A  
JZ \$RCHAR1

DI  
PUSH D  
DCR M  
MOV D, M  
INX H  
MOV A, M  
CPI 1730  
JNC \$RC1.5  
CPI 1410  
JC \$RC1.5  
SUI 400

\$RCHARA EQU \*-1

\$RC1.5 PUSH PSW

\$RCHAR2 DCR D  
JM \$RCHAR3  
INX H  
MOV A, M  
DCX H  
MOV M, A  
INX H  
JMP \$RCHAR2

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

```
$RCHAR3  EI
          POP      PSW
          POP      D
          POP      H
          RET

**          $WCHAR - WHITE SINGLE CHARACTER

$WCHAR.   PUSH PSW
$WCHAR1   LDA      $CSLCTL
          ERRNZ     CC.HLD-1
          RAR
          JC        $WCHAR1
          RAR
          ERRNZ     CC.DMP-2
          JNC       $WCHAR2
          POP      PSW
          RET

*          OVERLOWN TYPE-AHEAD BUFFER, ECHO BELL.

$CSI2     DCR      M
          MVI      A, BELL
          PUSH PSW

*          TYPE CHARACTER

$WCHAR2   CALL $COSI
          ERRNZ     USR.TXR-1
          RAR
          JNC       $WCHAR2
          POP      PSW
          JMP      $CDOUT

**          $PRSC - PRESET CONSOLE DRIVERS

$PRSC.    MVI      A, 201A
          CALL     $CISO
          CALL     $TSOUT
          MVI      A, UCI.IR
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

```
CALL    $CISO
CALL    $TSOUT
MVI     A, UMI.1B+UMI.L8+UMI.16X
CALL    $CISO
CALL    $TSOUT
XRA     A
STA     $INBUF
MVI     A, 3030
STA     .UIVEC+6
LXI     H, $CSINT
SHLD    .UIVEC+7
MVI     A, UCI.ER+UCI.RE+UCI.IE+UCI.TE
JMP     $CISO
```

\*\* \$CSINT -- CONSOLE INTERRUPT PROCESSOR

```
$CSINT  PUSH    H
        PUSH    PSW
        CALL    $CISI
        ANI     USR.RXR
        CNZ     $CSI1
$CSIX   POP     PSW
        POP     H
        EI
        RET
```

\*\* HAVE DATA INTERRUPT

```
$CSI1   LXI     H, $INBUF
        MVI     A, $INBUFL
        CMP     M
        CC      $CSI2
```

\* ADD CHARACTER TO QUEUE

```
$CSI3   INR     M
        MOV     A, M
        ADD     L
        MOV     L, A
        CALL    $CDIN
        ANI     1770
```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

```

        MOV        M,A

*        CHECK FOR SPECIAL CONTROL SEQUENCES

        CPI        / /
        RNC

*        HAVE CONTROL CHARACTER

$CSI5   MVI        L,##CSIB-2
        INX        H
        INX        H
        CMP        M
        RC
        INX        H
        JNE        $CSI5
        LDA        $CSLCTL
        ANA        M
        INX        H
        XRA        M
        STA        $CSLCTL
        PUSH       PSW
        MVI        L,##INBUF
        DCR        M
        POP        PSW
        RP

*        HAVE SPECIAL CONTROL CALL USER ROUTINE

        LHLD       $CSIC
        PCHL

        STL        'PAM/8 - ASSEMBLY CONSTANTS'

***      I/O PORTS

IP.PAD  EQU        3600
OP.CTL  EQU        3600
OP.DIG  EQU        3600
OP.SEG  EQU        3610
IP.TPC  EQU        3710
```



<H><U><G> <S><O><F><T><W><A><R><E> <V><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

OP.TPC	EQU	371Q
IP.TPD	EQU	370Q
OP.TPD	EQU	370Q

\*\* ASCII CHARACTERS

A.SYN	EQU	026Q
A.STX	EQU	002Q

\*\* FRONT PANEL HARDWARE CONTROL BITS

CB.SSI	EQU	00010000B
CB.MTL	EQU	00100000B
CB.CLI	EQU	01000000B
CB.SPK	EQU	10000000B

\*\* DISPLAY MODE FLAGS

DM.MR	EQU	0
DM.MW	EQU	1
DM.RR	EQU	2
DM.RW	EQU	3

\*\* TAPE EQUIVALENCES

RT.MI	EQU	1
RT.BP	EQU	2
RT.CT	EQU	3

\*\* BLOCK SIZE FOR INTER-PRODUCT COMMUNICATION.

BLKSIZ	EQU	512
--------	-----	-----

\*\* MACHINE INSTRUCTIONS

MI.HLT	EQU	01110110B
MI.RET	EQU	11001001B
MI.IN	EQU	11011011B
MI.OUT	EQU	11010011B
MI.LDA	EQU	00111010B
MI.ANI	EQU	11100110B

<H><U><G> <S><O><F><T><W><A><R><E> \* <U><O><L> \* II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

```

MI.LXID EQU      00010001B

**      USER OPTION BITS

UO.HLT EQU      10000000B
UO.NFR EQU      CB.CLI
UO.DDU EQU      00000010B
UO.CLK EQU      00000001B

**      8251 USART BIT DEFINATIONS
*
**      MODE INSTRUCTION CONTROL BITS
*
**      GIVEN IN CONSL
*
STL      'MAIN PROGRAM'

**      START OF CODE
*
*      PROGRAM ENTRY POINTS WITHIN PAM
*
SRS EQU      002265A
RNP EQU      002325A
RNB EQU      002331A
TPERR EQU      002205A
TFT EQU      002133A
CTC EQU      002172A

**      ROUTINE READT
*
*      READS ONE RECORD FROM TAPE INTO - INBUF
*
READT EQU      *
CALL      SRS      SCAN FOR REC START
MOV      L,A
SHLD     INBUFS    STORE COUNT
XCHG     (DE) = COUNT

```

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

*	LXI INX	H, INBUF H	(HL) = START BUF
RDT1	CALL MOV INX DCX MOV ORA JNZ	RNB M, A H D A, D E RDT1	READ BYTE PUT IN BUF  DECREMENT COUNT  MORE-GOTO RDT1
*	CALL	RNP	READ CRC SUM AND DISCARD IT
	XRA OUT RET	A OP, TPC	STOP TAPE RETURN TO CALLER
.START	EQU CALL	* \$PRSC	
	LXI	SP, ST.PTR	
	LXI INX MOV MOV CALL	H, INBUF H A, M C, A READT	READ LABEL RECORD
TOP	LXI INX MOV MOV CALL	H, INBUF H A, M C, A READT	READ NEXT RECORD
	LXI MVI INX	H, J M, 0 H	
TOP1	MVI	M, 0	
P.0	LXI MVI	H, I M, 0	
P.1	MVI LXI	A, 99 H, I	

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

	SUB	M
	JC	P.2
	MOV	C,M
	MVI	B,0
	LXI	H,LBUF1
	DAD	B
	XCHG	
	LXI	H,BLNK
	MOV	C,M
	MOV	A,C
	STAX	D
	LXI	H,I
	MOV	C,M
	MVI	B,0
	LXI	H,LBUF2
	DAD	B
	XCHG	
	LXI	H,BLNK
	MOV	C,M
	MOV	A,C
	STAX	D
	LXI	H,I
	MOV	C,M
	MVI	B,0
	LXI	H,LBUF3
	DAD	B
	XCHG	
	LXI	H,BLNK
	MOV	C,M
	MOV	A,C
	STAX	D
	LXI	H,I
	INR	M
	JNZ	P.1
P.2	LXI	H,KK
	MVI	M,0
P.2A	MVI	A,2
	LXI	H,KK
	SUB	M
	JC	P.10
	LXI	H,J

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

	MOV	C,M
	INR	L
	MOV	B,M
	LXI	H,INBUF
	DAD	B
	MOV	A,M
	SUI	0
	JNZ	P.3
	LHLD	J
	INX	H
	SHLD	J
P.3	LXI	H,J
	MOV	A,M
	INR	L
	MOV	B,M
	LXI	H,INBUFS
	SUB	M
	INR	L
	MOV	C,A
	MOV	A,B
	SBB	M
	JC	P.3AA
	CALL	READT
	LXI	H,J
	MVI	M,0
	INX	H
	MVI	M,0
P.3AA	LXI	H,J
	MOV	C,M
	INR	L
	MOV	B,M
	LXI	H,INBUF
	DAD	B
	MOV	A,M
	LXI	H,DOLS
	SUB	M
	JZ	P.10
	LXI	H,KK
	MOV	C,M
	MVI	B,0
	INR	L



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

	DAD	B	
	MOV	A,M	
	LXI	H,K	
	MOV	M,A	
P.3A	LXI	H,J	
	MOV	C,M	
	INR	L	
	MOV	B,M	
	LXI	H,INBUF	
	DAD	B	
	MOV	A,M	
	LXI	H,SLSH	
	SUB	M	
	JZ	P.5	
	LXI	H,J	
	MOV	C,M	
	INR	L	
	MOV	B,M	
	LXI	H,INBUF	
	DAD	B	
	MOV	A,M	
	MOV	C,A	
	MUI	A,2000	
	SUB	C	
	JNC	P.4	
	LXI	H,J	
	MOV	C,M	
	INR	L	
	MOV	B,M	
	LXI	H,INBUF	
	DAD	B	
	XCHG		
	LXI	H,BLNK	
	MOV	C,M	
	MOV	A,C	
	STAX	D	
P.4	LXI	H,K	
	MOV	C,M	
	MUI	B,0	
	LXI	H,LBUF1	
	DAD	B	

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

	PUSH	H
	LXI	H, J
	MOV	C, M
	INR	L
	MOV	B, M
	LXI	H, INBUF
	DAD	B
	MOV	A, M
	POP	H
	MOV	M, A
	LHLD	J
	INX	H
	SHLD	J
	LXI	H, K
	INR	M
	JMP	P. 3A
P. 5	LHLD	J
	INX	H
	SHLD	J
	LXI	H, KK
	MOV	C, M
	MVI	B, 0
	INR	L
	DAD	B
	MOV	A, M
	LXI	H, K
	MOV	M, A
P. 5A	LXI	H, J
	MOV	C, M
	INR	L
	MOV	B, M
	LXI	H, INBUF
	DAD	B
	MOV	A, M
	LXI	H, SLSH
	SUB	M
	JZ	P. 7
	LXI	H, J
	MOV	C, M
	INR	L
	MOV	B, M

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

	LXI	H, INBUF
	DAD	B
	MOV	A, M
	MOV	C, A
	MVI	A, 2000
	SUB	C
	JNC	P. 6
	LXI	H, J
	MOV	C, M
	INR	L
	MOV	B, M
	LXI	H, INBUF
	DAD	B
	XCHG	
	LXI	H, BLNK
	MOV	C, M
	MOV	A, C
	STAX	D
P. 6	LXI	H, K
	MOV	C, M
	MVI	B, 0
	LXI	H, LBUF2
	DAD	B
	PUSH	H
	LXI	H, J
	MOV	C, M
	INR	L
	MOV	B, M
	LXI	H, INBUF
	DAD	B
	MOV	A, M
	POP	H
	MOV	M, A
	LHLD	J
	INX	H
	SHLD	J
	LXI	H, K
	INR	M
	JMP	P. 5A
P. 7	LHLD	J
	INX	H

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

P.7A

SHLD	J
LXI	H, KK
MOV	C, M
MVI	B, 0
INR	L
DAD	B
MOV	A, M
LXI	H, K
MOV	M, A
LXI	H, J
MOV	C, M
INR	L
MOV	B, M
LXI	H, INBUF
DAD	B
MOV	A, M
LXI	H, SLSH
SUB	M
JZ	P.9
LXI	H, J
MOV	C, M
INR	L
MOV	B, M
LXI	H, INBUF
DAD	B
MOV	A, M
MOV	C, A
MVI	A, 2000
SUB	C
JNC	P.8
LXI	H, J
MOV	C, M
INR	L
MOV	B, M
LXI	H, INBUF
DAD	B
XCHG	
LXI	H, BLNK
MOV	C, M
MOV	A, C
STAX	D

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

P.8	LXI	H,K
	MOV	C,M
	MVI	B,0
	LXI	H,LBUF3
	DAD	B
	PUSH	H
	LXI	H,J
	MOV	C,M
	INR	L
	MOV	B,M
	LXI	H,INBUF
	DAD	B
	MOV	A,M
	POP	H
	MOV	M,A
	LHLD	J
	INX	H
	SHLD	J
	LXI	H,K
	INR	M
	JMP	P.7A
P.9	LHLD	J
	INX	H
	SHLD	J
	LXI	H,KK
	INR	M
	JNZ	P.2A
P.10	LXI	H,CR
	MOV	A,M
	CALL	\$WCHAR
	LXI	H,I
	MVI	M,0
P.10A	MVI	A,99
	LXI	H,I
	SUB	M
	JC	P.11
	MOV	C,M
	MVI	B,0
	LXI	H,LBUF1
	DAD	B
	MOV	A,M



<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

	CALL	\$WCHAR
	LXI	H, I
	INR	M
	JNZ	P.10A
P.11	LXI	H, LF
	MOV	A, M
	CALL	\$WCHAR
	LXI	H, ESC
	MOV	A, M
	CALL	\$WCHAR
	INR	L
	MOV	A, M
	CALL	\$WCHAR
	LXI	H, BLNK
	MOV	A, M
	CALL	\$WCHAR
	LXI	H, I
P.12	MVI	M, 0
	MVI	A, 99
	LXI	H, I
	SUB	M
	JC	P.13
	MVI	A, 99
	SUB	M
	INR	L
	MOV	M, A
	MOV	C, M
	MVI	B, 0
	LXI	H, LBUF2
	DAD	B
	MOV	A, M
	CALL	\$WCHAR
	LXI	H, I
	INR	M
	JNZ	P.12
P.13	LXI	H, CR
	MOV	A, M
	CALL	\$WCHAR
	INR	L
	MOV	A, M
	CALL	\$WCHAR

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER < CONT'D >

P. 14	LXI	H, I
	MVI	M, 0
	MVI	A, 99
	LXI	H, I
	SUB	M
	JC	P. 15
	MOV	C, M
	MVI	B, 0
	LXI	H, LBUF3
	DAD	B
	MOV	A, M
	CALL	\$WCHAR
	LXI	H, I
	INR	M
P. 15	JNZ	P. 14
	LXI	H, CR
	MOV	A, M
	CALL	\$WCHAR
	INR	L
	MOV	A, M
	CALL	\$WCHAR
	CALL	\$WCHAR
	CALL	\$WCHAR
	CALL	\$WCHAR
	LXI	H, J
	MOV	C, M
	INR	L
	MOV	B, M
	LXI	A, INBUF
	DAD	B
	MOV	A, M
	LXI	H, DOLS
	SUB	M
	JNZ	P. 16
	EI	
P. 16	HLT	
	LXI	H, J
	MOV	A, M
	INR	L
	MOV	B, M
	LXI	H, INBUFS

<H><U><G> <S><O><F><T><W><A><R><E> <U><O><L> II

PROGRAM NAME: MAIL LABEL - FARMER &lt; CONT'D &gt;

SUB	M
INR	L
MOU	C, A
MOU	A, B
SBB	M
JNC	TOP
JMP	P. 0
EI	
HLT	

```

*** SPACE AND DATA CONSTANTS

```

ORG 060000A

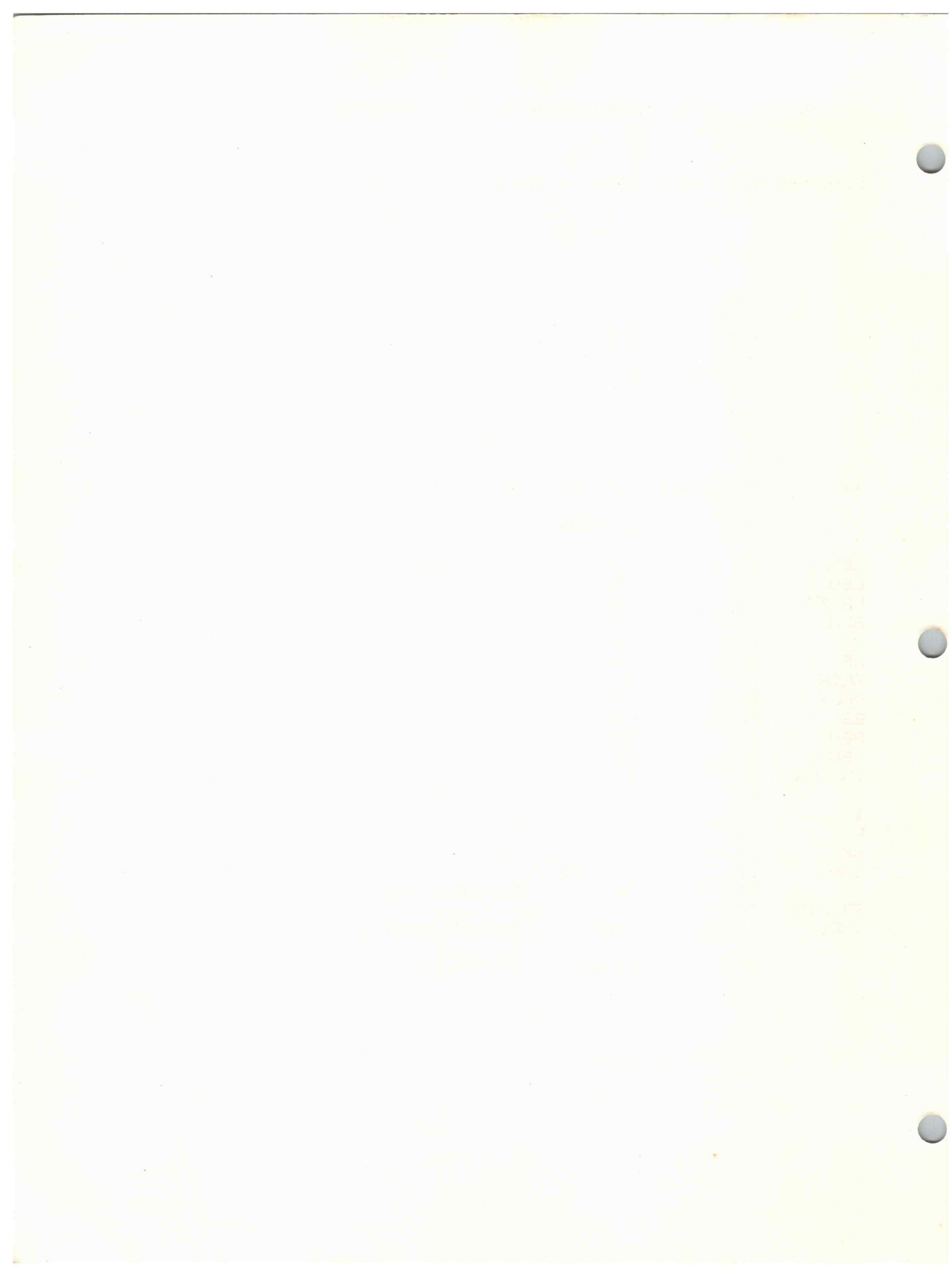
```

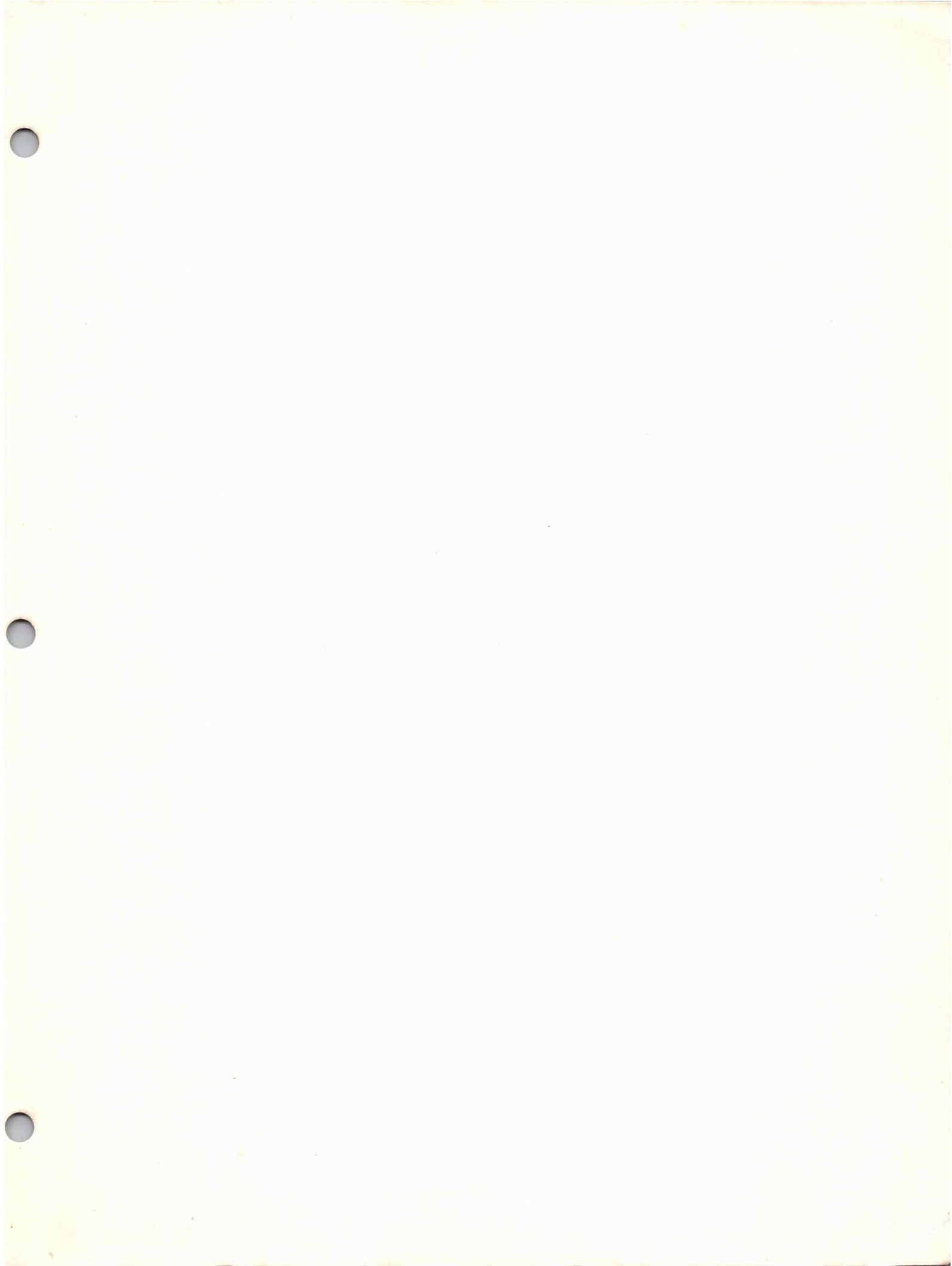
LBUF1      DS      150
LBUF2      DS      150
LBUF3      DS      150
BLNK       DB      20H
CR         DB      0DH
LF         DB      0AH
SLSH       DB      2FH
DOLS       DB      24H
ESC        DB      1BH
REV        DB      36H
LBUFS      DB      99
INBUFS     DS      2
J          DS      2
I          DS      1
K          DS      1
KK         DS      1
INDX       DB      1,37,74
           DS      50             DEFINE STACK
ST.PTR     DS      2
INBUF      DS      1000         DEFINE INPUT BUF

END      ENTRY

```

06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----







[illegible]